Ipso Facto

The A-C-E Magazine

July, 1980

INDEX PAGE

ISSUE # 18

IPSO FACTO is published by the ASSOCIATION OF COMPUTER EXPERIMENTERS (A.C.E), a non-profit, educational organization.

Send ALL A.C.E. correspondence to Bernie Murphy,
102 McCraney St.,
Oakville, Ontario,
Canada. L6H 1H6

ASSOCIATION OF COMPUTER EXPERIMENTERS 1980-81 EXECUTIVE

PRESIDENT:	John Norris	3 Country Club Toronto, Ont. 416-239-8567
PAST President:	Ken Bevis	220 Cherry Post Rd., Mississauga, Ont., L5A 1H9 416-277-2495
SECRETARY/ Treasurer:	Mike Franklin	24 Duby Rd., Acton, Ont., L7J 2P1 519-853-3421
EDITOR:	John VanLierde	40 Arlington Ave., Toronto, Ont., M6G 3K8 416-656-3185
ASSISTANT EDITOR:	John Myszkowski	99 Augusta St., Hamilton, Ont., 416-529-0250
	Fred Feaver	105 Townsend Ave., Burlington, Ont., L7T 1Y8 416-637-2513
CONSULTANT:	Bob Silcox	562 Forestwood Cr., Burlington, Ont., L7L 4K3 416-845-1630
DRAUGHTSMAN:	John Myszkowski	
MEMBERSHIP COORDINATOR:	Bernie Murphy	102 McCraney St., Oakville, Ont., L6H 1H6 416-845-1630
	Don MacKenzie	3124 Bonaventure Dr., Mississauga, Ont., L4T 2J2 416-676-9084
PROGRAM COORDINATOR:	Jeff Davis	8 Hillview Dr., Grimsby, Ont., L3M 4E5
TRAINING COORDINATOR:	Fred Feaver	
	Ken Bevis	
SOFTWARE COORDINATOR:	Wayne Bowdish	149 East 33rd. St., Hamilton, Ont., L8V 3T5 416-388-7116

A.C.E. EXECUTIVE

continued on Page 10 ...

EDITORIAL COMMENTS

Great news !!!!

We are pleased to announce that we are now in a position to take orders for not only De Facto (as promised last time), but also for three club-designed boards.

D E F A C T O

This book collects in one volume all of the best material in issues 1-18, with all known errata corrections incorporated into the articles, many diagrams improved, and a comprehensive table of contents. It is, therefore, not just of interest to those who are missing back issues.

De Facto is unbound, but is three-hole punched to fit standard binders, comes in a plastic shrink wrapping, and has a three-colour cover.

The price for De Facto is as follows:
- to Canadian addresses -- $15 Canadian funds.
- to U.S.A. addresses ---- $18 American funds.
- to overseas addresses -- $20 American funds. (sent airmail)

De Facto is a truly great collection of 1802-related material, and, at the price, the best bargain around.

BOARDS

1) 8K EPROM Board (as depicted in issue 17, and described in issue 16).
2) Kluge Board (also depicted in issue 17).
3) Club Backplane (also depicted in issue 17).

The list of parts for the EPROM board was given with the article on it in issue 16, page 28.

All boards are sold as bare boards. (schematics and parts lists enclosed).

The price for each board is
--to Canadian addresses - $20 Canadian.
--to U. S. and overseas - $20 American.

All orders (boards and De Facto) should be sent to:

Bernie Murphy
102 McCraney Street
Oakville, Ontario
Canada. L6H 1H6

EXTRA ! See pg. 32 for Colour Video Board offer !

>

A CHEAP PRINTER

R. N. Thornton
1403 Mormac Road
Richmond, Va. 23229

I recently decided to add a printer to my micro, a home-brew ELF based on the Popular Electronics articles. Naturally, I wanted manuscript quality, quiet operation, low power, variable paper width up to 14 inches, tractor and/or pinch feed, high reliability, etc., etc. After writing to a number of suppliers, I found that such a printer would cost roughly 20 times my total computer investment to date. Even a used Selectric typewriter was $350 without interface. Finally, I saw an advertisement in ON-LINE (now the Computer Shopper) for a 32 character per line printer for $95. This represented quite a compromise relative to my original desires, but was affordable. After further consideration, I realized the printer would be a perfect companion for a 32 character per line video display. I ordered the printer from:

Carl Poulson
1601 Roder Ct.
Streamwood, Illinois 60103

sending my money order for $95.00 (U.S.). The printer arrived in about 3 weeks via U.P.S. It was well packed, and weighed about 45 pounds. The printer is a used, reconditioned model made by Bunker Ramo, and contains a dot matrix print mechanism made by the Victor Comptometer Company. The cabinet is about 18"x16"x6½", and contains the power supply, printer mechanism, and electronics for the ASCII interface. The cabinet top lifts for easy maintenance access, and all parts are fastened with twist lock screws for fast removal. The printer mechanism is a 5x7 dot matrix mechanism which prints 80 characters per second in 32 columns on 3.5 inch wide adding machine paper. A complete set of schematics and adjustment procedures is supplied with the printer.

The ASCII interface uses 7-Bit ASCII in parallel. To use the printer, the buffer must be loaded with 32 characters, one at a time, and then a print command is issued.

To load a character into the buffer, the character is presented in parallel on the data lines, and the print command line is brought high. The computer must then wait until the printer busy line goes high and back to low before loading the next character. After all 32 characters are loaded into the buffer, an ASCII carriage return (hex ØD) is placed on the data lines, and the print command line is brought high. Loading for the next line may begin after the print busy line goes high and then low. Since the print mechanism works right to left, if fewer than 32 characters are loaded, they will be right justified on the line. The print character set includes Ø-9, A-Z (upper case only), blank, and 19 special characters. A 2516 ROM is used to generate the print characters. I believe a 2708 EPROM could be used to change/expand the character set, though I haven't tried this yet.

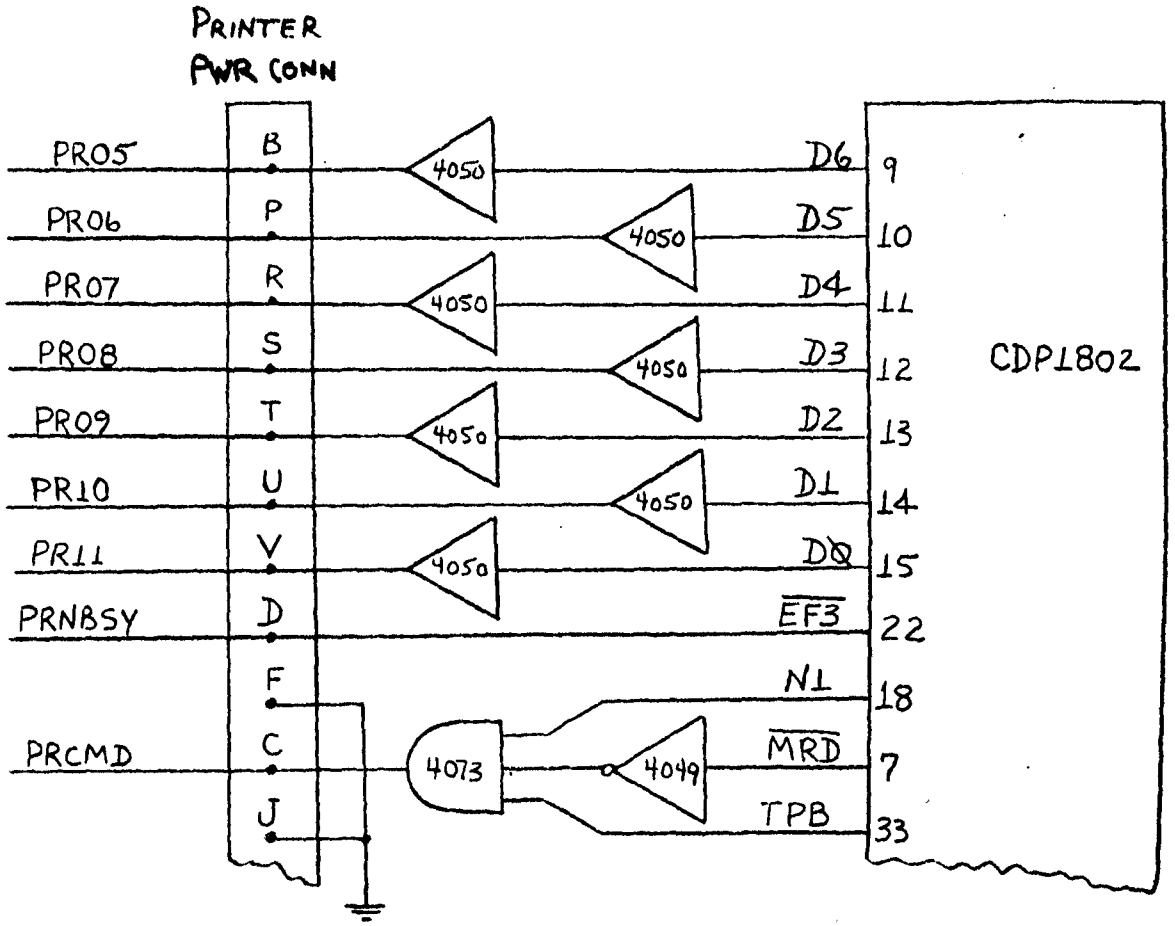
```
----------PRINT SAMPLE----------

                 0123456789ABCDEF
 HEX 20-2F:  .'#$%&@( )*+,-./
 HEX 30-3F: 0123456789$-+.
 HEX 40-4F: .ABCDEFGHIJKLMNO
 HEX 50-5F: PQRSTUVWXYZ:\=↑?
```

I used the following routine to test printer operation. With some modification, it should be usable as a print subroutine.

LOC	HEX	LABEL	MNEMONIC	REMARKS
ØØ	9Ø		GHI	D=Ø
Ø1	B3B4		PHI,PHI	R3.1=Ø, R4.1=Ø
Ø3	F81EA3	START	LDI,PLO	R3=DATA ADDRESS
Ø6	E3		SEX	X=3
Ø7	F82ØA4		LDI,PLO	R4=32, LOAD COUNTER
ØA	62	LOAD	OUT2	LOAD DATA TO BUFFER
ØB	36ØB	WAITBSY	B3 WAITBSY	LOOP UNTIL PRINTER BUSY
ØD	3EØD	WAITIDL	BN3 WAITIDL	LOOP UNTIL PRINTER IDLE
ØF	24		DEC	DECREMENT COUNTER
1Ø	84		GLO	D=LOAD COUNT
11	3AØA		BNZ LOAD	CONTINUE LOAD IF NONZERO
13	F81DA3		LDI,PLO	R4=C.R. ADDRESS
16	62		OUT2	PRINT THE LINE
17	3617	WAITPRT	B3 WAITPRT	LOOP UNTIL PRINTER BUSY
19	3E19	WAITFRE	BN3 WAITFRE	LOOP UNTIL PRINTER IDLE
1B	3ØØ3		B START	REPEAT
1D	ØD	CR		CARRIAGE RETURN
1E		DATA		32 BYTES OF PRINT DATA

This routine is written to mate with the hardware interface used: if a different N-line is used for print command, the OUT2 instructions must be changed; if $\overline{EF3}$ is not used for printer busy, the B3 and BN3 instructions must be changed. The instruction at 1B will have to be changed to a valid exit except for testing. As written, the 32 bytes located at 1E-3D will be printed over and over.

The hardware interface I used is shown below. It was necessary to buffer the data lines, as the printer uses TTL logic. I had trouble with the interface until reading the RCA 1802 manual which clearly stated the requirement for AND'ing N1, MRD, and TPB to insure memory data is available when the print command line is brought high.



Dear Bernie:

We've had several articles on EPROM Programmers, but nothing on erasers. There is an EPROM Eraser kit available for $25.00 (U.S.) from:

R. W. Electronics, Inc.
3203 North Western Avenue
Chicago, Illinois 60618

The kit includes a G.E. Ultraviolet bulb #G4T4.1, lamp socket, switch, and G.E. ballast #89G489. Instructions included with the parts show how to build the eraser. Additional parts required include a line cord, bread pan, and small parts which are easily fabricated. A 2708 can be erased in 15-20 minutes with the assembled eraser.

The best buy I've seen on 2708's is $6.95 for the 450 NSEC version from:

Active Electronic Sales Corp.
P. O. Box 1035
Framingham, Mass. 01701

My thanks to all on the Executive Staff who produce this outstanding newsletter.

Sincerely,
Dick Thornton
1403 Mormac Road
Richmond, Virginia 23229

ENHANCEMENTS TO UT4

Frederick K. Hannan
10 Filosi Road
East Lyme, Connecticut 06333

Shortly after Mike Franklin's article on the UT4 program appeared in IPSO FACTO, he was kind enough to send me a tape of his program. Subsequently, I purchased the RCA Manual MPM 224 and found both to be very valuable tools.

However, after using the UT4 program for some time, I found it contained some illogical, impractical, or just plain dumb commands and routines. In order to fit it to my own personal quirks, I have made some changes to the program which might be of use to others.

First, although I did not experience any problems with the TIMALC routine, I thought it was best to load the time constant into RE.1 after initilization, similar to the manner suggested by Dave Taylor in IPSO FACTO #17. This will help overcome any future problems as equipment ages, etc.

I also felt it was rather dumb to put my machine through the REGISTER SAVE routine when there was no valid reason for doing this. Consequently, I established a new initialization routine at the beginning of UT4, again, identical to Dave Taylor's routine on Page 42, IPSO FACTO #17.

The result of this is to free up addresses XX16 through XX38 for other changes or new routines (XX is high order address of UT4).

My next thought was to bring some sense and logic to the three command symbols. After all, !M and $P do not really represent their true functions of Writing to memory (!) or Running program ($). Only ?M comes close to its function of Dump memory.

My solution was to use "W" for Write to memory, "D" for Dump memory and "R" for Run program. This is accomplished by the following changes:

Address	XX4A	from FB24 to FB 52
	XX52	from FB1E to FB13

Further, I felt it was superfluous to have to enter the character "M" or "P" when entering a command. The changes to eliminate the "M" and "P" are:

Address	XX4C	from32D6 to 32DB
	XX56	from D3 to 305B

Now the three commands are:

W XXXX Δ Data - Write to memory
D XXXX Δ # of Bytes - Dump memory
R XXXX - Run program

The second change was to add a space after the prompt symbol and to change the prompt symbol from "*" to "⟩" (my personal favorite). The change of the prompt symbol is easily accomplished by substituting 3E for the 2A at address XX41.

I had a little difficulty in getting a space after the prompt, however. At first I merely moved the beginning of the "START" routine from location XX39 to XX37 and added "D320" to the end of the prompt print string. For some reason, I have yet to figure out, this would not work. My final fix was to add another TYPE routine pointer after the first two "D3XX" SEP SUBS.

Thus, my start is now:

```
Address  XX34   F89CA3   TYPE routine pointer
         XX37   D30D     TYPE CR
         XX39   D30A     TYPE LF
         XX3B   F89CA3
         XX3E   D33E     TYPE ⟩
         XX40   D320     TYPE space
```

The branch instruction at address XX14 must be changed to 3034 for the new "START" address. Also, the RESTART branch instructions at addresses XX99 and XXB9 should be changed to 3234.

Lastly, I felt it was rather stupid to have to manually input a LINE FEED after the use of a COMMA/CR in the Write to memory mode. After all, our machines are pretty smart and can do this kind of simple task with ease.

In order to make any changes in the area of the program concerned with the COMMA input, some room was needed to fit in a subroutine to automatically insert a CR/LF after a comma was typed.

Initially, I thought I could free enough space by moving the SYNTAX error routine from address XXCA through XXd1 to the now free area at address XX2B through XX33. Although this did not give me enough room for my subroutine, I proceeded with the relocation of the error routine as it gave me the opportunity to consolidate the present two routines (XXCA to XXD1 and XYF8 to XYFE) into one. Also, by placing the routine immediately ahead of the START routine, the ERROR routine would naturally fall through to the START routine. These changes are:

```
Address  XX2B   F89CA3   TYPE routine pointer
         XX2E   D30D     TYPE CR
         XX30   D30A     TYPE LF
         XX32   D33F     TYPE ?
```

Several branch instructions must also be changed as follows:

```
Address  XX59  from 3ACA to 3A2B
         XX63  from 3ACA to 3A2B
         XX75  from 3ACA to 3A2B
         XXAF  from 3BCA to 3B2B
         XXD9  from 3ACA to 3A2B
         XXE0  from 3ACA to 3A2B
```

Now, to automatically insert the CR/LF after a comma in the Write to memory routine, I inserted a new subroutine at XX16 as follows:

```
Address  XX16 F89CA3 TYPE routine pointer
              D30D   TYPE CR
              D30A   TYPE LF
              F83BA3 Set subroutine pointer to READ AH
              F8ABA5 Set main PC for return
              D5     Return
```

Address XXBD should be changed to 3216. The program will now branch to the new subroutine on input of a comma, do a CR/LF and return to address XXAB, ready for more input.

This completes my present modifications to UT4. A future change will be the insertion of a counter into the Write to memory routine to automatically perform a CR/LF after every 16 bytes of input and completely eliminate the COMMA routine. Right now, I am busy using UT4 to help in implementing a Serial I/O interface for "The Monitor" presented by Steve Nies in a past IPSO FACTO. (Note to Steve - by all means, keep us advised of your changes to your monitor.)

Finally, I must express my feelings about Netronic's Full Basic and Quest's Super Basic.

Although I run Tiny Basic, Full Basic, and Super Basic, I find that Super Basic, in spite of its length, offers so much more versatility that Tiny and Full are virtually unused. The availability of the two dimension arrays in Super is a prime requisite for me.

Also, perhaps it is ignorance on my part, but I find the steps necessary to perform a very complex equation in RPN is a painful and time consuming task. Although I leave the Math Board in my ELF II permanently (I use the empty 16 pin socket locations for termination points for the interconnecting cables to my backplane.), it is very seldom used. In fact, if it did not offer 8K of Eprom sockets, I would leave it out altogether.

With regard to the LOAD problem with Full Basic, I have installed a switch on my Giant Board which allows me to jumper bypass the mods required by the Math Board (Full Basic will run without the mods if no

math function is required.) For normal use of the Giant Board CASSETTE LOAD or Full Basic "LOAD", I close the switch. If I am RUNning Full Basic, the switch is opened.

The switch is a simple SPST slide switch Super-Glued to the top left corner of my Giant Board. I drilled a small hole in the plastic cover to accomodate a push/pull wire (bent paperclip). The switch reconnects the cut foil trace to restore the Giant Board to its original configuration. When the switch is opened, the Math Board modifications take effect.

Postscript

After writing this, I decided to change the SEMI-COLON input in the same manner as the COMMA input. This was easily accomplished by adding the following to the previous changes:

Address XXC3	F89CA3	Type Routine Pointer
	D30D	Type CR
	D30A	Type LF
	F83BA3	Set Subroutine Pointer to READAH
	F85BA5	Set Main PC for Return
	D5	Return
	C4	NOP

This change is not listed in the summary.

CONTINUED FROM PAGE 2

HARDWARE COORDINATOR:	Anthony Tekatch	RR 1 Caistor Centre, Ont., L0R 1E0 416-957-7556
HARDWARE PRODUCTION AND SALES:	Fred Pluthero	1013 Upper Wellington St., Hamilton, Ont., L9A 3S4 416-389-4070
PUBLISHING COMMITEE:	Dennis Mildon	44 Wildewood Ave., Hamilton, Ont., L8T 1X3 416-385-0798
	John Hanson	955 Harvey Place, Burlington, Ont., L7T 3E9 416-637-1076

SUMMARY OF UT4 CHANGES

ADDRESS	NEW DATA	(XX High Order Address)
XX00	F8XXB3B5	Relocate "Start" routine - See
04	F808A5	Dave Taylor letter, Ipso Facto
07	D5	#17, Page 42
08	E5	
09	7155	
0B	6101	
0D	F8FEA3	
10	D3	
11	F824BE	
14	3034	
XX16	F89CA3	Comma/CR/LF Subroutine
19	D30D	
1B	D30A	
1D	F83BA3	
20	F8ABA5	
23	D5	
XX2B	F89CA3	Syntax Error Routine
2E	D30D	
30	D30A	
32	D33F	
XX34	F89CA3	Initialize Routine
37	D30D	
39	D30A	
3B	F89CA3	
3E	D33E	
40	D320	
XX4A	FB52	Change of Command Symbols
52	FB13	($ now R, : now W, ? now D)
XX4C	32DB	New Branches to Eliminate need of
56	305B	"P" and "M"
XXBD	3216	Branch to Comma Subroutine
XX59	3A2B	New Syntax Error Branches
63	3A2B	
75	3A2B	
AF	3B2B	
D9	3A2B	
E0	3A2B	
XX99	3234	New Restart Branches
B9	3234	

A. D. Barksdale Garbee
1601 Clayton Avenue
Lynchburg, VA 24503

ANOTHER STEPPER

After reading the article by Mr. Airhart in issue #9, I decided to submit the enclosed circuit, which I have been using for quite some time.

IC1, a 555 timer chip, is wired to run as an astable oscillator. Speed variation is accomplished with R2, which is a trimmer in order to save space.

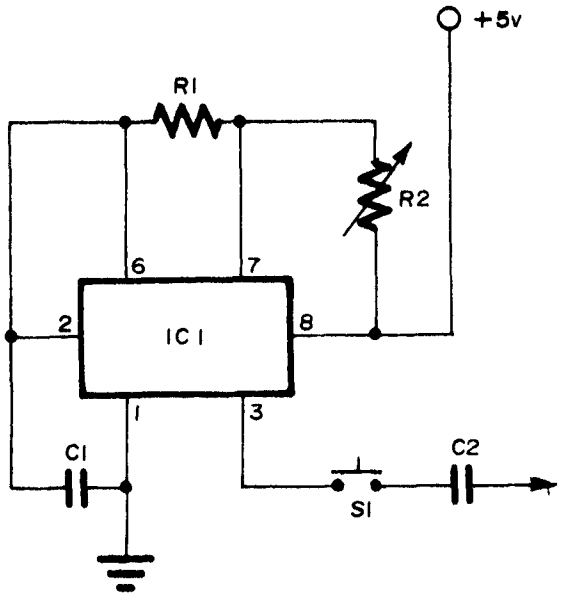
My computer, a Quest Super Elf, already has provision for stepping through memory when the computer is in LOAD, by repeatedly pushing the Input button. Therefore I simply connected the output of this circuit to the input switch. This and the variable resistor for speed are the main differences between my design and that of Mr. Airhart.

To use the circuit on a Super Elf, press Reset, Load, Memory Protect, and the flip the toggle switch. When the desired location approaches, flip the toggle switch again. Then press the Input switch manually to reach the exact location. To alter memory, step to the location before that which is to be changed, press the Wait button, and proceed to load data normally.

Some Super Elfs may require a buffer on the output of the circuit. I doubt if this device could be used at all by anyone with a TEC unit, but those who own the Super Elf will find it an indespensible device.

REFERENCES:

1. Super Elf Users Manual
2. Radio Shack 555 data sheet



INPUT PULSER DIAGRAM

PARTS LIST

IC1	555 timer chip
C1	.2 mfd
C2	.1 mfd
R1	5K
R2	500K trimmer
S1	SPST toggle

Richard N. Thornton
1403 Mormac Road
Richmond, Va. 23229

AUDIBLE CONTINUITY TESTER

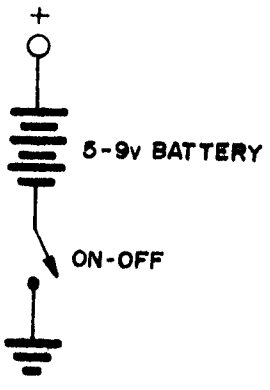
This device can be used to test continuity, and provide some indication of the amount of resistance in a circuit. In operation, the probes are touched to the points to be tested. If the circuit is open, the speaker is silent. If there is continuity, even at high resistance (10 megohms or more), a tone will be heard. The tone decreases in frequency as resistance between the probe tips increasew. A noticeable decrease in frequency can be noted at resistances as low as 1,000 ohms.

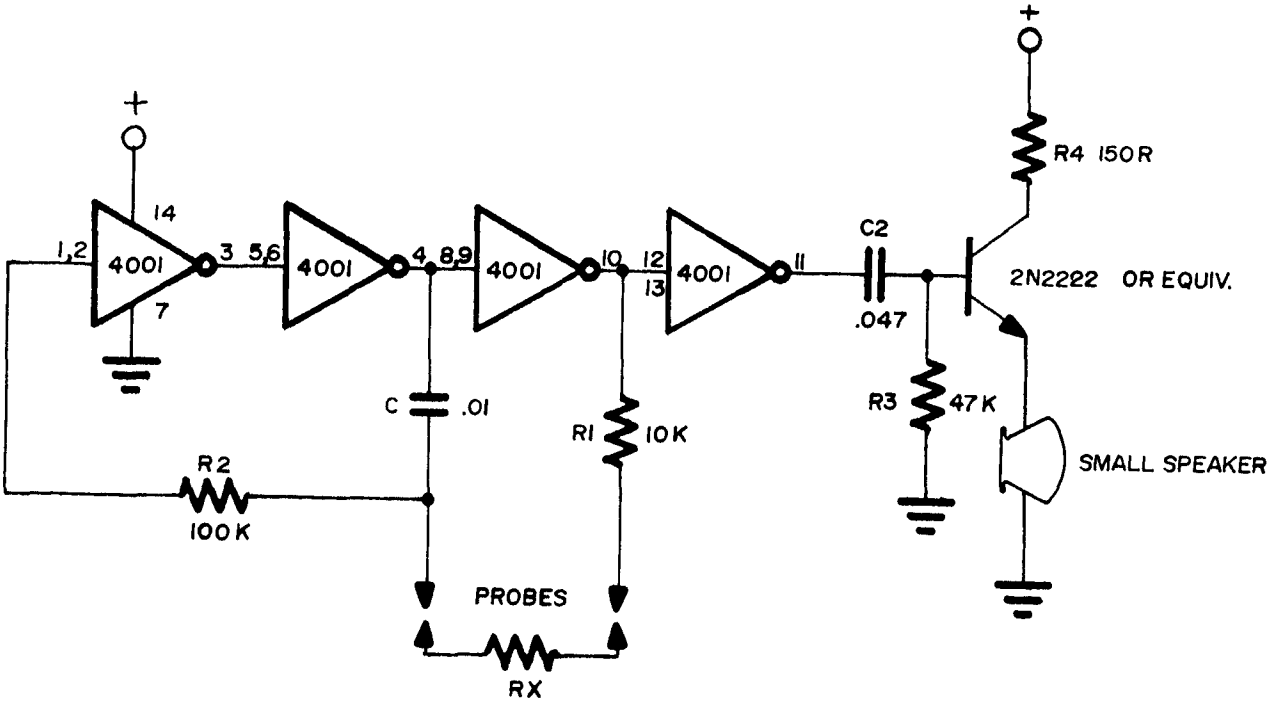
As drawn below, the audible tone is about 5kHz with the probes shorted. If a lower frequency is desired, increase the value of R1, or play with the values of C and R2. To increase the volume, reduce the value of R4, or increase the value of C2 and reduce R3. I use a 9 V transistor radio battery, but the circuit works well at 5 V. With probes disconnected, power supply drain is not measurable with a 0-50 mA meter. With probes shorted, the current is about 20 mA.

The circuit is essentially that shown in National Semiconductor's "CMOS DATA BOOK" from AN-118 "CMOS Oscillators", where the frequency is given as

$$f \cong \frac{1}{2C(R1+RX)\left(\frac{.405R2}{R1+RX+R2}+.693\right)}$$

Observations: The probes appear to be open (no tone) when connected across a diode, or a capacitor of less than about .04 uF. Capacitors greater than .04 uF produce a tone similar to a short circuit. With the probes connected to the input to my oscilloscope, I see a differentiated square wave with about 30-35 V peak-to-peak. When the 'scope input is DC, the pulses are continuous. When the 'scope input is switched to ac, the pulse train dies out in about 15 seconds, but repeats if the meter leads are reversed. I don't understand these things, just report them. Perhaps someone more knowledgeable can supply some insight. The high voltage pulses might damage sensitive devices, but don't seem to harm the 4001 IC in the circuit.





GAMES 1802s PLAY!

P. Thyssen,
Julianastead 39,
5951CH Belfeld,
Netherlands (Limburg)

The following are several games I have just written. The Orthogame is the first I have ever written in Assembler so be careful with it! Till yet, I am just writing games for fun. I've also a lot of Tiny Basic programs but most of them have a Dutch text. The nicest ones are:

+ a word guessing game (2K RAM like WORD in David Ahl's 101 BASIC programs
+ Coldik. Escape from a war prisoner's camp. This one I wrote after analyzing the Quest game (The first is in Dutch but the second has the same text as in Byte, July 1979). A problem is that I have them only on tape. They require at least 4-5K RAM without Tiny Basic.

Maybe someone can rewrite the Dutch games. If someone sends me one or both items below, I'll send him my games on cassette tape. I'll pay also for the copy cost by sending a post check:

(1) The RCA CHIP-8 language and copy of programs and user manual. I've tried to get it in Holland and wrote RCA (USA) but never heard anything.

(2) The PILOT 2K language (also terminal and tape routines) and copy of user manual for my ELF II and video display (Netronics).

RUSSIAN ROULETTE:

```
00   90 B1 B2 B3 B8
05   B9 BF
07   F8 2F A3
0A   F8 C7 A2          -¦
0D   F8 13 A1           ¦
10   D3                 ¦
11   72                 ¦
12   70                 ¦
13   22 78              ¦
15   22 52              ¦
17   C4 C4 C4           ¦    DMA transfer TV
1A   F8 00 B0           ¦
1D   F8 00 A0           ¦
20   80 E2              ¦
22   E2 20 A0           ¦
25   E2 20 A0           ¦
28   E2 20 A0           ¦
2B   3C 20              ¦
2D   30 11              ¦
                        -¦
2F   E2 69              MAIN;
31   F8 80 AF           (SUBF)
34   9F B4              M(C3): COUNTER


36   F8 C3 A4           M(C4): RND NUMBER
                        -¦
39   F8 01 54            ¦  TIME
3C   F8 50 B6            ¦       DELAY
3F   26                  ¦
40   96                  ¦
41   3A 3F               ¦
                        -¦
43   E4 64              DISPLAY NUMBER ON LEDS
45   9F DF              ??
47   3F 47
49   37 49
4B   F8 01 DF
4E   04
4F   FE FE
51   F4                 Rn = Rn-1 * 5 + 1   'till input key = 1
52   FC 01
54   54
55   37 59
57   30 4E
59   FF D6              > 216 HIT!!!
5B   33 67
5D   24
5E   04 FC 01 54         COUNTER + 1
62   F8 02 DF           KhK!
65   30 3C              AGAIN
67   F8 03 DF           bANG!!
```

```
6A   F8 DB A8
6D   F8 80 58                    ¦ BULLET
70   08 F6 58                    ¦
73   32 31                       AGAIN
75   F8 0F B6                    -¦
78   26                            ¦ TIME
79   96                            ¦      DELAY
7A   3A 78                       -¦
7C   30 70
7E   00
7F   D3
80   FE                          SUBF:
81   FC 9B A8
84   F8 D4 A9                    -¦
87   08 59                         ¦  CONVERSION TO
89   18 19                         ¦
8B   08 59                         ¦    TV-DISPLAY
8D   88 FC 07 A8
91   89 FC 07 A9                 -¦
95   FB FC
97   3A 87
99   30 7F
9B   E0 E0 AE E0 A8 8A 77 77
A3   20 20 CA A0 C8 0C 55 55
AB   40 40 8E E0 88 88 77 57
B3   00 00 CC C0 C8 8C 45 51
BB   40 40 AA A5 AE 8A 45 57
D2   80                           -¦
DA   7F                           ¦
E2   70                           ¦  PISTOL
EA   C0                           ¦
F2   C0                          -¦
```

KALEIDOSCOPE:

```
00   90 B3
02   F8 83 AF                 (SUBF) -¦
05   F8 6D AE                 (SUBE)  ¦
08   F8 2D A3                 (MAIN)  ¦
0B   F8 6B A2                 (STACK) ¦
0E   F8 14 A1                 (CNTT)  ¦
11   D3                               ¦
12   72                               ¦  DMA TRANSFER
13   70                               ¦
14   22 78                            ¦  FOR TV
16   22 52                            ¦
18   C4 C4 C4                         ¦
1B   93 B0 A0                         ¦
1E   80 E2                           ¦
20   E2 20 A0                         ¦
23   E2 20 A0                         ¦
26   E2 20 A0                         ¦
29   3C 1E                            ¦
2B   30 12                           ¦
2D   E2 69                          -¦
```

```
2F   E9
30   F8 10 A6
33   00
34   26
35   86
36   3A 33
38   37 38
3A   F8 67 A9
3D   14
3E   84
3F   3A 42
41   DE
42   DE
43   AA A5
45   8D AB
47   19
48   DE AC
4A   DF 8D DF
4D   8B A5 8C DF 8D DF
53   8C A5 8A DF 8B DF
59   8D A5 8A DF 8B DF
5F   30 30
61   00 00 00 00 00 00
67   RND1
68   RND2
69   00 00 00
6C   D3
6D   09
6E   FE FE
70   F4
71   FC 01
73   59
74   A6
75   FF 05
77   33 74
79   AD
7A   86 FD 04
7D   30 6C
7F   86
80   F3
81   59
82   D3
83   FC 08 A7
86   FE FE FE
8A   FC D2 A9
8D   87
8E   A8
8F   19
90   88
91   FF 08
93   33 8E
95   F8 80
97   A6
98   88
99   32 7F
9B   28
9C   86 F6
9E   30 97
```

```
-¦
 ¦ TIME DELAY
 ¦
-¦

STOP!

COMPUTE: Y-X: AA
         X-5: AB
         Y-Y: AC
         Y-5: AD

  BACK

random seed value (1 byte)
random seed value (1 byte)
  (STACK)

 Rn MOD 5

ad <=> x-5
D <=> y-x

LOOK AT GIVEN BIT BY R7 AND R9
IF BIT EQUALS ZERO MAKE 1
IF BIT EQUALS 1 MAKE ZERO

RETURN
```

Flip/Flop.

The object of this game is to change a row of eight X's (XXXXXXXX) to a row of eigth O's (OOOOOOOO).
By typing the desired rownumber plus Input-key you can change an X to O (or reverse),but it will also change another position too.
Can you figure out a minimum strategy?
User remarks:

1) Set/reset RUN and the Q-led goes on. Tip the Input-key and the line of X's will appear on the bottom of the TVscreen.
2) To reset the line to all X's (same game!) type ≠00. To start a new game (new set flip/flops) type a value greater than ≠08 plus twice the Input-key.
3) The hexadecimal Led's will display the number of guesses.
4) When you've got all O's the Q-led goes on. The program will now reset like typing ≠00.
5) If there is no solution(?) use twice the same number. The second time you'll get another flip/flop.

The secret of this program lies in the "nasty" property of pseudorandomgenerators, that they always need a starting value. Each starting value correspond to one set of flip/flops, so totally you'll have about 256 play variants. For more change the randomgeneratorroutine.
Much XOXXOOXO.

Literature: David Ahl, Basic Computer Games. 1978,page 63

```
00  90 B1 B2 B3
04  B4 B5 BB BC
08  BD BE BF
0B  F8 AD AF  (subF)
0E  F8 B8 AE  (subE)
11  F8 36 A3  (main)
14  F8 AB A2  (stack)
17  F8 1D A1  (intrpt)
1A  D3
1B  72 70
1D  22 78
1F  22 52
21  C4 C4 C4
24  9F B0 A0
27  80 E2
29  E2 20 A0
2C  E2 20 A0
2F  E2 20 A0
32  3C 27
34  30 1B
36  E2 69
38  F8 A5 A4  (RND)
3B  7B
3C  DF
3D  3F 3C
3F  AA       :starter
40  7A
41  F8 A6 A5:counter
44  9F 55 A6 A8
48  86
49  FC E8 AB: load row
4C  F8 1C 5B  of X's
4F  16
50  86 DE
52  86
53  FF 08
55  3B 48
57  37 57
59  E5 64
5B  3F 5B
5D  37 5D
5F  31 40    :reset Q on
61  6C
62  32 40    :reset same
64  FF 09      game
66  33 3B    :reset new
68  05 DE      flip/flops
6A  88 F3
6C  3A 72
6E  88 FE A8
71  C8
72  05 A8
74  A7
75  8A 54    :compute other
77  DF         flip/flop
78  27
79  87
7A  3A 77
7C  04
7D  A7       :mod9
7E  87
7F  FF 09
81  33 7D
83  E5
84  87
85  32 6E
87  F3
88  32 6E
8A  87 DE
8C  05 A8
8E  25
8F  05
90  FC 01 55: counter+1
93  F8 E8 AB: test for
96  0B          all 0's
97  FB 1C
99  3A 59
9B  1B
9C  8B
9D  FF F0
9F  3B 96      :if yes
A1  7B           Q on
A2  30 59
A5  RND
A6  counter
AB  stack
AC  D3
AD  E4 04   : subroutine
AF  FE FE     pseudorandom
B1  F4        Rn=Rn-1x5+1
B2  FC 01
B4  54
B5  30 AC
B7  D3
B8  FC E7 AB: subroutine X→ 0
BB  FC 08 AC                0→ X
BE  FC 08 AD
C1  0B
C2  FB 1C
C4  3A CF
C6  F8 14 5B 5D :load X
CA  F8 08 5C
CD  30 B7
CF  F8 1C 5B 5D :load 0
D3  F8 14 5C
D6  30 B7
remain zero
```

--ORTHO--

The game was originally invented by the mathematician Stanislaw Ulam. On may points there is correspondence with the Game of Life, but while Life simulates the complex world of living organism, imitates Ortho growth. In nature we find frequently that this happens through symmetrical ground patterns.(like crystals) The subject of this game is an orthogonal symmetry.

Definition:
- You play on a grid of squares or cells(chessboard).
- A cell has finite states of appearence(bl/wh,colors)
- In the beginning of the game you can occupy cells or not.
- Any further course is fully determined through a defined set of playing rules.
  Ortho: In the next generation is always that cell occupied, which has perpendicular only one neighbour. All cells of generation n dies, when generation n+i appears. So, there are always i generations visible.

There is chosen for a 15x15 matrix, but theoretical there will be a continued growth on an infinite grid. An advantage of Ortho as caleidoscope are the many starting possibillities, but the nicest results you will get with symmetrical patterns. In the figure you see the development of one point for i=2. Each generation could be represented by a color, but if you have only black/white TV?
The program is written in Assembler and will run on a ELFII, 1K RAM and 1861 TV-chip.
Some suggestions: In M(00B9) the value ≠03 is similar to i=1, ≠04 to i=2, ≠ with i=3 etc.
In M(00A2) you could change the condition "one neighbour".

User descriptions: If you set/reset RUN the Q-Led goes on. Put in the desired coordinates (Y,X from 0..E,≠23 is Y=2 and X=3) followed by the Input-key, and the cell will appear on the TV-screen. If there was already a cell on that place it will disappear, so errors could be corrected. The hexadecimal Led's will display your inputvalue.
You can go on until the value is greater than ≠EE. Then the game will start and the computed patterns will appear on the screen.
If you want a new game set/reset RUN and....

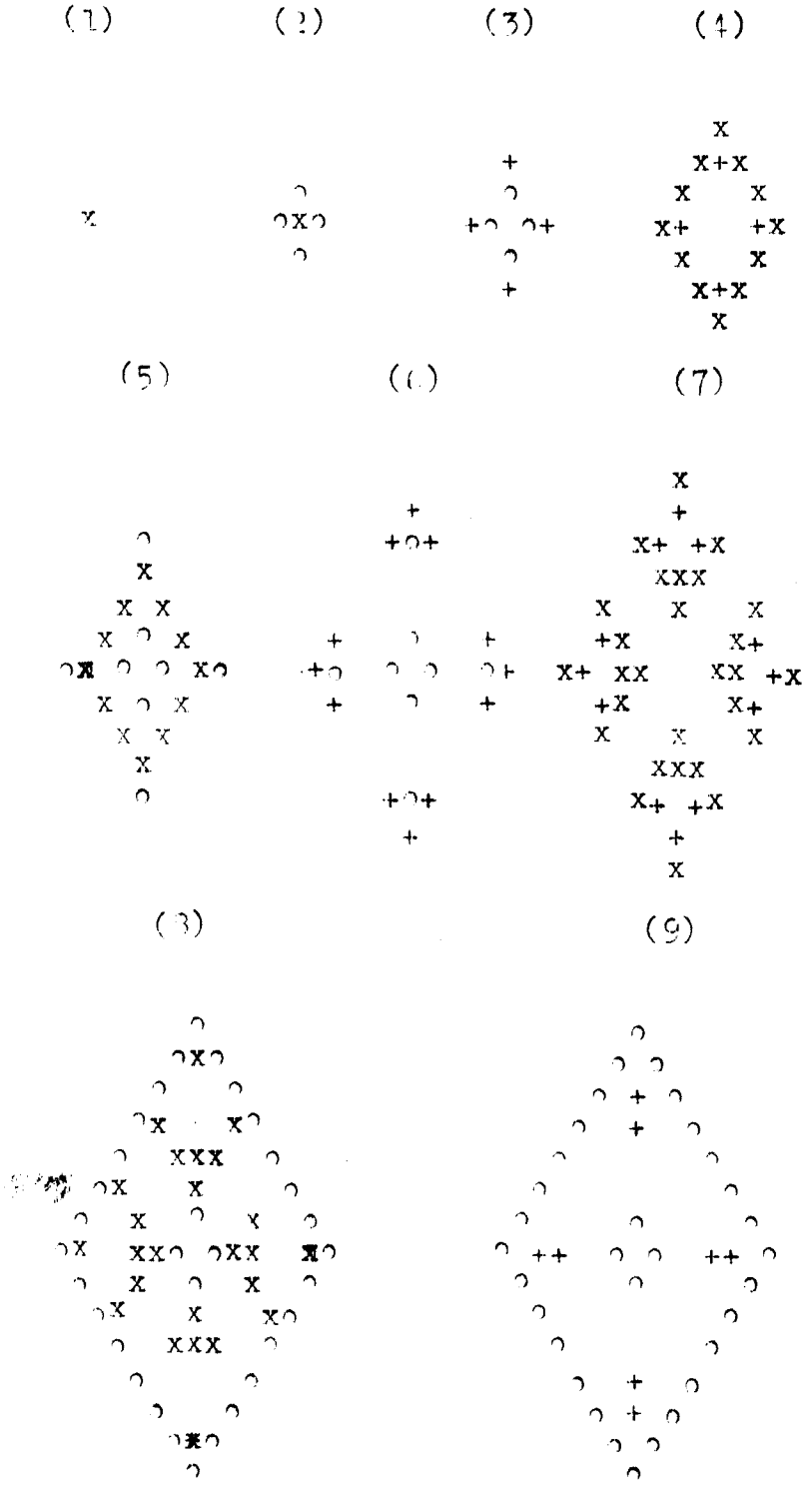


figure: Development of one point for i=2.

SYMBOL TABLE.

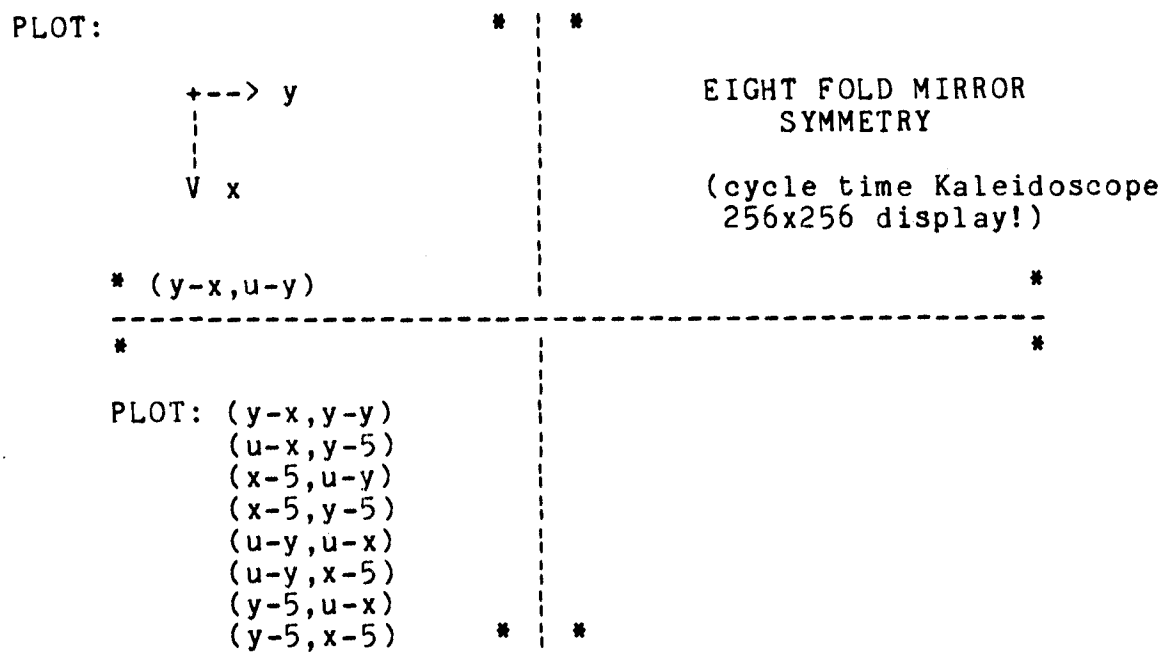
```
0000 DMA      0000 START    00BF LOOP2
0001 INT      001C RETURN   00D7 STACK
0002 SPR      001E INTRPT   00D8 GNRTIE
0003 MPC      002B REFRSH   00D9 RETMPC
0004 MEM      0046 MAIN     00DA COMPTE
0005 GNE      0050 LOOP     0100 TERUG
0006 VRX      005D INPUT    0107 QUIT
0007 VRY      0072 BEGIN    010D RTNMPC
0008 CNT      0079 LOOP1    010E PLOT
0009 TMP      008A OUT1     0112 MOD16
000A RGT      0090 OUT2     011D MOD4
000C TVS      0098 OUT3     0126 SHIFT
000D CEL      00A0 OUT4     0200 TVSCRN
000E CPE      00A7 BACK     0300 MEMORY
000F PLT      00B5 CYCLUS
```

OBJECT LIST.

```
0000          ..--ORTHO--
0000          ..
0000          ..P.THIJSSEN
0000          ..BELFELD
0000          ..NETHERLANDS
0000          ..
0000          ..REGISTERS
0000          ..
0000 DMA=0    ..DMA pointer
0000 INT=1    ..interrupt
0000 SPC=2    ..stack pointer
0000 MPC=3    ..main prog. counter
0000 MEM=4    ..dataarea
0000 GNE=5    ..adres gen. counter
0000 VRX=6    ..counter X
0000 VRY=7    ..counter Y
0000 CNT=8    ..counter neighbours
0000 TMP=9    ..workregister
0000 RGT=10   ..workregister
0000 TVS=12   ..TV-area
0000 CEL=13   ..examined cell
0000 CPE=14   ..sub "compute"
0000 PLT=15   ..sub "plot"
0000          ..
0000          ORG Ø
0000          ..initialization
0000 90B1     START: GHI DMA;PHI INT
0002 B2B3            PHI SPR;PHI MPC
0004 B5BE            PHI GNE;PHI CPE
0006 F801            LDI A.1(PLOT)
0008 BF              PHI PLT
0009 F80E            LDI A.0(PLOT)
000B AF              PLO PLT
000C F81E            LDI INTRPT
000E A1              PLO INT
000F F8D7            LDI STACK
0011 A2              PLO SPR
0012 F846            LDI MAIN
0014 A3              PLO MPC
0015 F8D8            LDI GNRTIE
0017 A5              PLO GNE
0018 F8DA            LDI COMPTE
001A AE              PLO CPE
001B D3              SEP MPC
001C            ..interrupt routine
001C 7270     RETURN:LDXA;RET
001E 2278     INTRPT:DEC SPR;SAV
0020 2252            DEC SPR;STR SPR
0022 C4              NOP
0023 C4              NOP
0024 C4              NOP
0025 F802            LDI A.1(TVSCRN)
0027 B0              PHI DMA
0028 F800            LDI A.0(TVSCRN)
002A A0              PLO DMA
002B 80E2     REFRSH:GLO DMA;SEX SPR
002D E220            SEX SPR;DEC DMA
002F A0              PLO DMA
0030 E220            SEX SPR;DEC DMA
0032 A0              PLO DMA
0033 E220            SEX SPR;DEC DMA
0035 A0              PLO DMA
0036 E220            SEX SPR;DEC DMA
0038 A0              PLO DMA
0039 E220            SEX SPR;DEC DMA
003B A0              PLO DMA
003C E220            SEX SPR;DEC DMA
```

```
003E A0          PLO DMA
003F E220        SEX SPR;DEC DMA
0041 A0          PLO DMA
0042 3C2B        BN1 REFRSH
0044 301C        BR  RETURN
0046         ..clear memory+tvscreen
0046 E269 MAIN:  SEX SPR;INP 1
0048 F803        LDI A.1(MEMORY)
004A B4          PHI MEM
004B F802        LDI A.1(TVSCRN)
004D BC          PHI TVS
004E 92A4        GHI SPR;PLO MEM
0050 AC   LOOP:  PLO TVS
0051 9254        GHI SPR;STR MEM
0053 5C          STR TVS
0054 14          INC MEM
0055 84          GLO MEM
0056 3A50        BNZ LOOP
0058 F803        LDI A.1(MEMORY)
005A B4BD        PHI MEM;PHI CEL
005C         ..input data
005C 7B          SEQ
005D 3F5D INPUT: BN4 x
005F 375F        B4  x
0061 E2          SEX SPR
0062 6CA4        INP 4;PLO MEM
0064 6422        OUT 4;DEC SPR
0066 FFEF        SMI #EF
0068 3372        BPZ BEGIN
006A E4          SEX MEM
006B F801        LDI #01
006D F754        SM;STR MEM
006F DF          SEP PLT
0070 305D        BR INPUT
0072         ..compute new generation
0072 F802 BEGIN: LDI #02
0074 55          STR GNE
0075 3775        B4  x
0077 92A7        GHI SPR;PLO VRY
0079 92A6 LOOP1: GHI SPR;PLO VRX
007B 7B          SEQ
007C DEAD        SEP CPE;PLO CEL
007E 0D          LDN CEL
007F 3AA7        BNZ BACK
0081 7A          REQ
0082 92A8        GHI SPR;PLO CNT
0084 87          GLO VRY
0085 328A        BZ  OUT1
0087 27          DEC VRY
0088 DE17        SEP CPE;INC VRY
008A 86   OUT1:  GLO VRX
008B 3290        BZ  OUT2
008D 26          DEC VRX
008E DE16        SEP CPE;INC VRX
0090 87   OUT2:  GLO VRY
0091 FF0E        SMI 14
0093 3398        BPZ OUT3
0095 17          INC VRY
0096 DE27        SEP CPE;DEC VRY
0098 86   OUT3:  GLO VRX
0099 FF0E        SMI 14
009B 33A0        BPZ OUT4
009D 16          INC VRX
009E DE26        SEP CPE;DEC VRX
00A0 88   OUT4:  GLO CNT
00A1 FB01        XRI #01
00A3 3AA7        BNZ BACK
00A5 055D        LDN GNE;STR CEL
00A7 16   BACK:  INC VRX
00A8 00          IDL
00A9 86          GLO VRX
00AA FF0F        SMI 15
00AC 3B7B        BL  LOOP1+2
00AE 17          INC VRY
00AF 87          GLO VRY
00B0 FF0F        SMI 15
00B2 3B79        BL  LOOP1
00B4         ..next generation?
00B4 05          LDN GNE
00B5 FC01 CYCLUS:ADI #01
00B7 55          STR GNE
00B8 FF04        SMI #04
00BA 33B5        BPZ CYCLUS
00BC         ..plot generations
00BC 7A          REQ
00BD 92A4        GHI SPR;PLO MEM
00BF FC02 LOOP2: ADI #02
00C1 AC          PLO TVS
00C2 E5          SEX GNE
00C3 04          LDN MEM
00C4 F3          XOR
00C5 3AC9        BNZ x+4
00C7 9254        GHI SPR;STR MEM
00C9 925C        GHI SPR;STR TVS
00CB DF          SEP PLT
00CC 14          INC MEM
00CD 84          GLO MEM
00CE 3ABF        BNZ LOOP2
00D0 F803        LDI A.1(MEMORY)
00D2 B4          PHI MEM
00D3 3075        BR  BEGIN+3
00D5         ..
00D5         ORG x+2
00D7 00      STACK:,0
00D8 00      GNRTIE:,0
00D9         ..compute+testroutine
00D9 D3      RETMPC:SEP MPC
00DA E2      COMPTE:SEX SPR
00DB 87          GLO VRY
00DC FEFE        SHL;SHL
00DE FEFE        SHL;SHL
00E0 52          STR SPR
00E1 86          GLO VRX
00E2 F4          ADD
```

```
00E3 A4           PLO MEM
00E4 31D9         BQ  RETMPC
00E6 E5           SEX GNE
00E7 04           LDN MEM
00E8 32D9         BZ  RETMPC
00EA F3           XOR
00EB 32D9         BZ  RETMPC
00ED 18           INC CNT
00EE 30D9         BR  RETMPC
00F0      PAGE
0100      ..plotroutine
0100 EC   TERUG:  SEX TVS
0101 3907         BNQ QUIT
0103 8A           GLO RGT
0104 F3           XOR
0105 300C         BR RTNMPC-1
0107 04   QUIT:   LDN MEM
0108 320D         BZ  RTNMPC
010A 8A           GLO RGT
010B F4           ADD
010C 5C           STR TVS
010D D3   TRNMPC: SEP MPC
010E F8F9 PLOT:   LDI #F9
0110 AC           PLO TVS
0111 84           GLO MEM
0112 A9   MOD16:  PLO TMP
0113 8C           GLO TVS
0114 FC08         ADI 8
0116 AC           PLO TVS
0117 89           GLO TMP
0118 FF10         SMI 16
011A 3312         BPZ MOD16
011C 89           GLO TMP
011D A9   MOD4:   PLO TMP
011E 1C           INC TVS
011F 89           GLO TMP
0120 FF04         SMI 4
0122 331D         BPZ MOD4
0124 F8C0         LDI #C0
0126 AA   SHIFT:  PLO RGT
0127 89           GLO TMP
0128 3200         BZ  TERUG
012A 29           DEC TMP
012B 8A           GLO RGT
012C F6F6         SHR;SHR
012E 3026         BR SHIFT
0130      PAGE
0200      TVSCRN:,0
0201      PAGE
0300      MEMORY:,0
0301      END
```



>

A Hardware Bug in the 1802

G. Pick
Box 1023,
Botwood, Nfld.

If your system uses interrupts and you are also outputting data using R2 as your X register, you may find the data at M(R(X)) destroyed under certain circumstances.

The following section of a telemetry program brought this problem to my attention.

```
00    Bsel     DB       Call hex keybd inp. sub.
01              32       B2 exit -- exit this function
02             [  ]      if input = '00'
03              FD       SDI
04              02       02 -- Do not accept input over 02
05              3B       BNF
06             [00]      Bsel
07    Alarm     63       Out 3  Output bank select data
08              22
09              22       Dec. R2  Point R2 to clear area
0A              6B       Input 3  Input alarm data
0B              64       Output alarm data to display
0C              22       Dec R2
0D              F8 00    LDI 00
0F              52       Str R2 - output 00 to reset alarms
10              63       Out 3
11              37       B4 - Br. to input rtn. if 'input' on
12             [00]      Bsel
13              30
14             [07]      Br to Alarm - Output bank select again
```

The X register for this program was R2, and while it was running, an interrupt-driven real-time clock was also running, at 60 interrupts/second. When the hex keyboard data was entered the correct output data would be displayed for a fraction of a second and then would change to 00. A scope showed that output port 63 data was changing from X1 or X2 to X3 for no apparent reason. (63 is only a 4 bit output, so X means not significant).

After some chip replacement and thought, the reason was found, and was not a failure or direct program error.

In the 1802, an interrupt can occur between any 2 instructions, and output instructions are self incrementing. Therefore, if an interrupt occurs directly after an output instruction, as at address 07, the stack pointer (R2) now points 1 byte above the output data. The interrupt then decrements the stack pointer and stores T (the interrupted program's X and PC values, 23 in this case) in the stack position occupied by the (now deceased) output data. On the next trip around the loop, the output will be 23 and the program is shot.

Once known about, you can program around this bug, but I have not yet seen any warnings or information on this from RCA, who surely must know about it.

(Additional material on the subject of interrupt processing was presented in issue #4, p. 28, and #5, p. 44 - Editor)

>

FEELING POWERLESS ???

For sale, one power supply with the following specifications:

+ 600 v @ 1/4 amp military supply

+ 150 v bias supply

+ 5 and 6 v AC filament supply

This supply uses oil filled transformers and is loaded with all sorts of impressive front panel switches, dials, meters and etc. Asking $40 or best offer.

Contact: L. Dunlop,
Apt. 1501,
36 Torrance St.,
Burlington, Ont., Canada.
L7R 2R9

>

TO "VIP" AN ELF

Dave Taylor,
2114 Comm Sq.,
Box 5718,
APO San Francisco,
Calif. USA 96519

I performed minor surgery on my ELF II in the summer of last year in order to adapt it to the VIP format. Basically I readdressed the operating system so that it would fit into the first 4K of my RAM, then interfaced a Hex keyboard to one of my free bus connectors. This arrangement was recently modified to utilize the output port on Giant Board in order to open up the bus position. The keyboard I constructed was designed along the same lines as the RCA keyboard shown in the VIP users manual. I have enclosed a listing of the address changes I made and a drawing of the keyboard interface. I have also utilized the Chip 8 program written by Paul Meows, but I found it to be too slow when running programs which required a fast keyboard response (such as "Lunar Lander" in the VP-710 manual).

The key assignments for game #11 (VIP Card Match Game) are located at addresses 0375 through 0384 (Hex) at the end of the program listing. Just by looking at RCA's listing compared to the shuffling I did to match my keyboard arrangement, I can see no logical arrangement (but I know that it's there somewhere). My recommendation would be to start shoving in new values between 00 and 0F in those address locations and see what popped up on the screen. This method may seem unprofessional, but it is quick and easy.

Concerning one other thing in the VIP users manual, I found that the tape LED wired across the tape in the line, to be one of the most useful additions I have made to my ELF II when loading tapes which may contain several programs.

VIP Operating system changes:

address

8001 change to new first page of relocated program

8056 change to address of second page of relocated system

800A change to the output port instruction for port used (67 on Giant Board)

819C change to the output port instruction for port used (67 on Giant Board)

NOTE: Do not use the highest two pages of RAM when relocating this program as the highest page will be written over when the system is called, and the second highest page will be written over when the Chip 8 bumps the stack down one page.

Following entry of the relocated system, it can be accessed by entering a long branch (C0 xx 00 where xx is the high address of the first page of the system) using the ELF keyboard. Then hold the "C" key down on the VIP keyboard when flipping the RUN switch up. After entering the system, utilized the system instructions in the VIP users manual to key in or examine programs.

VIP Chip 8 Changes

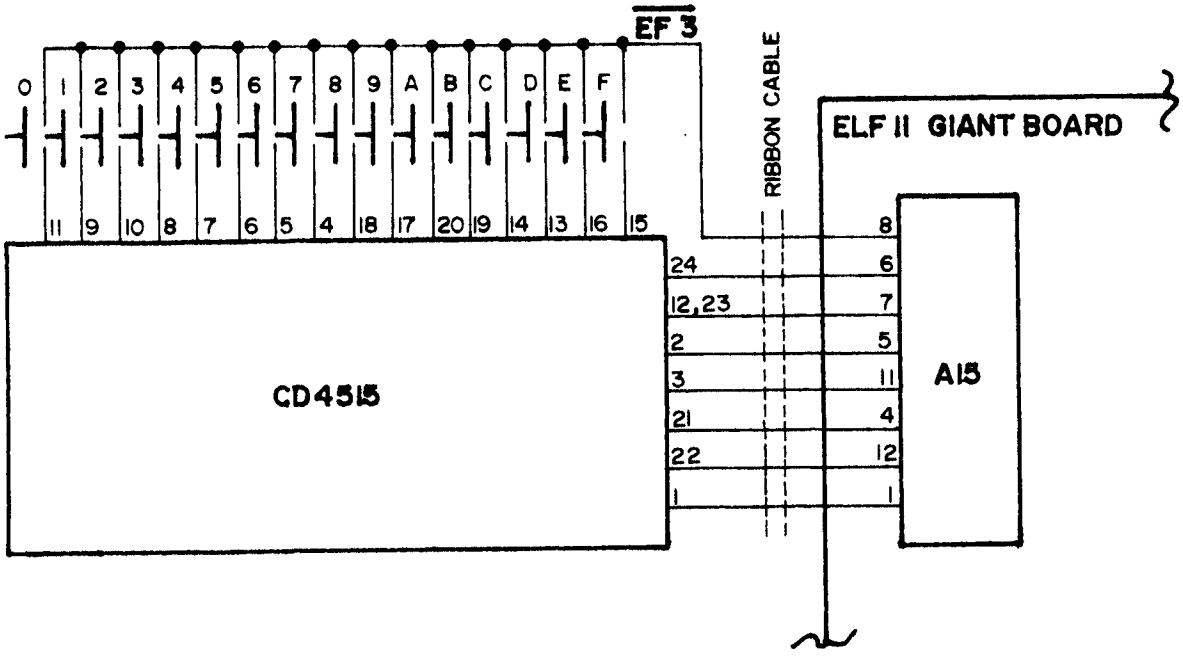
address

000A change to second page address of the relocated system

010B change to second page adsress of the relocated system

019A change to the output port instrcution for the port used (67 on Giant Board)

After loading a program witten in Chip 8 in pages 0200 to wherever, flip the RUN switch down and reload 9B B1 FF into the first 3 bytes of RAM. When the RUN switch is flipped up again, the VIP game should be running. For some reason, my ELF refused to run programs properly when I used the 91 BB FF in the users manual for the first 3 bytes. If a Chip 8 program is loaded on tape and then reloaded to the computer at a later date, the Operating System MUST be called up first to initialize the registers prior to running the Chip 8 program.



* Since +5v, GND and EF3 do not appear on A 15, they must be jumper from other locations on the Giant Board to unused pins on A 15.

>

SOME BASIC BUGS

Tom Pollard
Groton St.
Dunstable, Mass.
U.S.A. 01827

I have found some bugs in the netronics full basic.

1. Let command.
   Let (variable)=(expression)

   Try,

```
10 E=4
20 B=5
30 A=E
40 Print  A#B+
50 END
```

   A=the letter E not the variable E.
   Now change line 30 to :: 30 A=E#

2. INT command
   The problem is that when a negative Whole number is INT.
   The negative sine is lost.

   Try,

```
10 A=4.00009-
20 INT A
30 PR A
40 INT A
50 PR A
60 END
```

   Now add line 35

```
35 If A < 0 Let A=A#.0000001-
```

   Note only use up to 6 zeros

3. FOR/NEXT command
   When the NEXT is encountered the program must scan from the beginning to the next command. Not from the NEXT command to the beginning.
   Try,

```
10 FOR A= 1 TO 3
20 PR "LOOP 1"
30 NEXT A
40 FOR A= 1 TO 3
50 PR "LOOP 2"
60 NEXT A
70 END
```

```
40 FOR B=1 TO 3
60 NEXT B
```

   The only way to stop this is to use a different variable for line 40 and line 60

Maybe some one with a disassembled basic program could help the basic bugs.

Now for the HARD WARE.
In RADIO ELECTRONICS DEC 1978 a math board very close to netronics was shown. The clock for the math chip ran at 400KHZ. With a scope I found out that my clock ran at 200KHZ. So I ran this PRogram;

```
10 For A= 1 TO 1000
20 NEXT A
30 PR "END"
40 END
```

It took 6min. 44sec. So to get the oscilater up to 400KHZ. I changed C1 from 100pf to a 47pf. Now the program ran in 4min. 22sec.
Now with out a data sheet I kept on going untill I got a math error. When I did a math operation. So I went back to A 18pf cap. With a clock of 833KHZ. the time is now 3min. 10sec.
To see if your clock is running slow try the program. It should be about 4min. 22sec. If not get a hold of a scope and a junk box of caps.

MINI - RCABUG (Baudot Style)

Eric Chong
Noord Cura Cabai 49I
Box 151 - San Nicolas
ARUBA (Neth. Antilles)

After having read two fine articles on the UT-4 (Mike Franklin and Dave Taylor) serial monitor I worked them out and have ever since enjoyed the Tiny Pilot and Text Editor from Kilobaud.

Being a radio-amateur and still in the proud possession of an old Model-15 Teletype Printer (60 wpm) I would like to get my hands on a hex dump program in UT-4 style listing. With an awful slow printer a listing as UT-4 is the only way to speed up the dump a little.

So by combining two excellent articles (#I.F.10 RCABUG by Tom Crawford and I.F.6 Baudot Teletype by Brian Millier) and some small changes I brewed a mini-monitor with memory change and hex dump possibilities. I left the Cload and Cwrite subroutines out as the main purpose is machine dump listings.

The memory change routines are the same as RCABUG. The printed Dump routine has been modified to get a UT-4 style listing like:

```
Loc.  Data
0000  xxxx yyyy zzzz iiii aaaa bbbb cccc dddd
```

Also after each page an extra blank line is entered to seperate the

pages from each other. If you wish this can be changed:
1 page 2ebc FF, ½ page 2EBC 7F, ¼ page 2EBC 3F.
The GETHDR subroutine has been changed to first ask for the location and then how many bytes. This to keep up the tradition of UT-4.
Rest me only to thank Tom Crawford and Brian Millier for their wonderful ideas.

Memory map: Mini - Rcabug.

2E00 - 2E2D	..Init.	2E2E - 2E44	..Call
2E45 - 2E59	..Return	2E61 - 2E67	..BRKCHK
2E68 - 2E74	..PRTADR	2E75 - 2E7d	..2 x CRCRLF
2E7E - 2E86	..REQADR	2E87 - 2E98	..Input Byte
2E99 - 2ED3	..PRINTED DUMP	2ED4 - 2EF0	..Input Hex
2F00 - 2F07	..Outbyt 1A	2F08 - 2F12	..Build Address
2F13 - 2F24	..Outbyt 1B	2F25 - 2F2a	..1 x CRCRLF
2F2b - 2F36	..ASCII PRINT	2F37 - 2F5F	..GETHDR
2F60 - 2F6A	..messages	2F6B - 2F81	..Ineee
2F82 - 2FB8	..Outees (6)	2FB9 - 2FDE	..Print BAUDOT
2Fdf - 2FEC	..DELAY Baudot	2FF0 - 2FFF	..messages
3000 - 307F	..Look-up	3080 - 309E	..Mini Executive Loop
30A5 - 30CB	..Memory Change	30CC - 30D2	..QUEST
30ED - 30FF	..Stack area		

```
S
CD  LOC.: 2E00 BYTES: 0300

2F00  C490 B3E7 F808 A3D3 F824 A747 B547 A547
2E10  B447 A447 B247 A247 B147 A147 B047 A0E2
2E20  7AC0 3080 2E46 2E2F 30FF 2E00 00FD D3E4
2E30  7124 BF96 7386 7393 B683 A646 B346 A39F
2F40  F470 2430 2ED3 E571 25BF 96B3 86A3 E212
2F50  72A6 F0B6 9FE5 7025 3045 C02F 6B46 C02F
2F60  83FC 0037 67FF 00D5 99D4 2F00 89D4 2F00
2E70  D42F 8220 D5D4 2F25 D42F 25C0 2E99 D42F
2F80  2B2F F0D4 2F08 D5D4 2ED4 3398 FEFE FEFE
2E90  73D4 2ED4 6033 98F4 D5D4 2F37 D42F 25D4
2EA0  2F25 D42E 68D4 2E70 49D4 2F00 C4C4 2888
2EB0  3AB5 9832 C989 FA0F 3AC1 89FA FF3A 9F30
2EC0  9CF6 33A8 D42E 7030 A8D4 2F25 D42F 25D4
2ED0  2F25 30F1 D42E 5AFF 303B EEFF 0A3B E9FF
2EE0  073B EEFF 0633 FFFC 06FC 0AFC 00D5 FF00
2EF0  D536 F53F F16F FB0D 3AF1 C030 8000 0000

2F00  73D4 2F13 60F0 3017 D42E 8733 12B9 D42E
2F10  87A9 D5F6 F6F6 F6FA 0FF9 3052 FD39 02CF
2F20  FC07 C02F 83D4 2F2B 2FC6 D546 EB46 AB4B
2F30  322A D42F 5E30 2FD4 2E70 D42E 7E33 3789
2F40  A799 B7D4 2F2B 2FF7 D42F 0833 4389 A899
2F50  B887 A997 B989 5288 F499 5298 7433 43D5
2F60  0D0D 0A3F 0000 0D0D 0A00 0036 6F3F 6B6F
2F70  FEFE 3B7B FE3B 7BF8 20F5 38F0 D42F 83D5
2F80  C4C4 46AF FB20 32AF 8FFE FE3B A28C FBFF
```

```
2F90  32AE F87A D42F B930 AE8F D42F B9F8 00AC
2FA0  8FD5 8CFB 0032 99F8 6DD4 2FB9 3099 8FD4
2FB0  2FB9 F8FF AC30 A0C4 C4AE F830 BEC4 C40E
2FC0  FEFE 7BD4 2FDF FE32 D43B CF7A 30CF 7BD4
2FD0  2FDF 30C6 7AD4 2FDF D42F DFD5 C4C4 C4BC
2FE0  F8FC FF01 C4C4 C4C4 C43A E29C D500 0000
2FF0  4C4F 432E 3A20 0020 4259 5445 533A 2000

3000  3701 0101 0101 0129 0101 1101 0105 0101
3010  0101 0101 0101 0101 0101 0101 0101 0101
3020  092D 230B 2501 1735 3D13 0F17 0D31 0F2F
3030  1B3B 3321 1503 2E39 1907 1D1F 3D1D 1327
3040  0131 271D 2521 2D17 0B19 353D 130F 0D07
3050  1B3B 1529 0339 1F33 2F2B 2301 0101 0109
3060  0101 0101 0101 0101 0101 0101 0137 0101
3070  0101 0101 0101 0101 0101 3F01 0101 0101
3080  D42F 8207 D42F 25D4 2F82 3AD4 2E5A 73D4
3090  2E70 60F0 FF4D 32A5 FF44 CA2E 9930 CC01
30A0  0101 0101 01D4 2E7F 33CC D42F 25D4 2E68
30B0  09D4 2F00 D42F 70D4 2E5A FF20 3AC9 D42E
30C0  8733 CC59 5209 F53A CC19 30AA D42F 2B2F
30D0  6030 8401 0101 0101 0101 0101 0101 0101
30E0  0101 0101 0101 0101 0101 0101 019D 2F9D
30F0  2F7F 2FD7 2EB2 2F9D B204 742F C72E 8E30
```

A. C. E. Colour Video Board

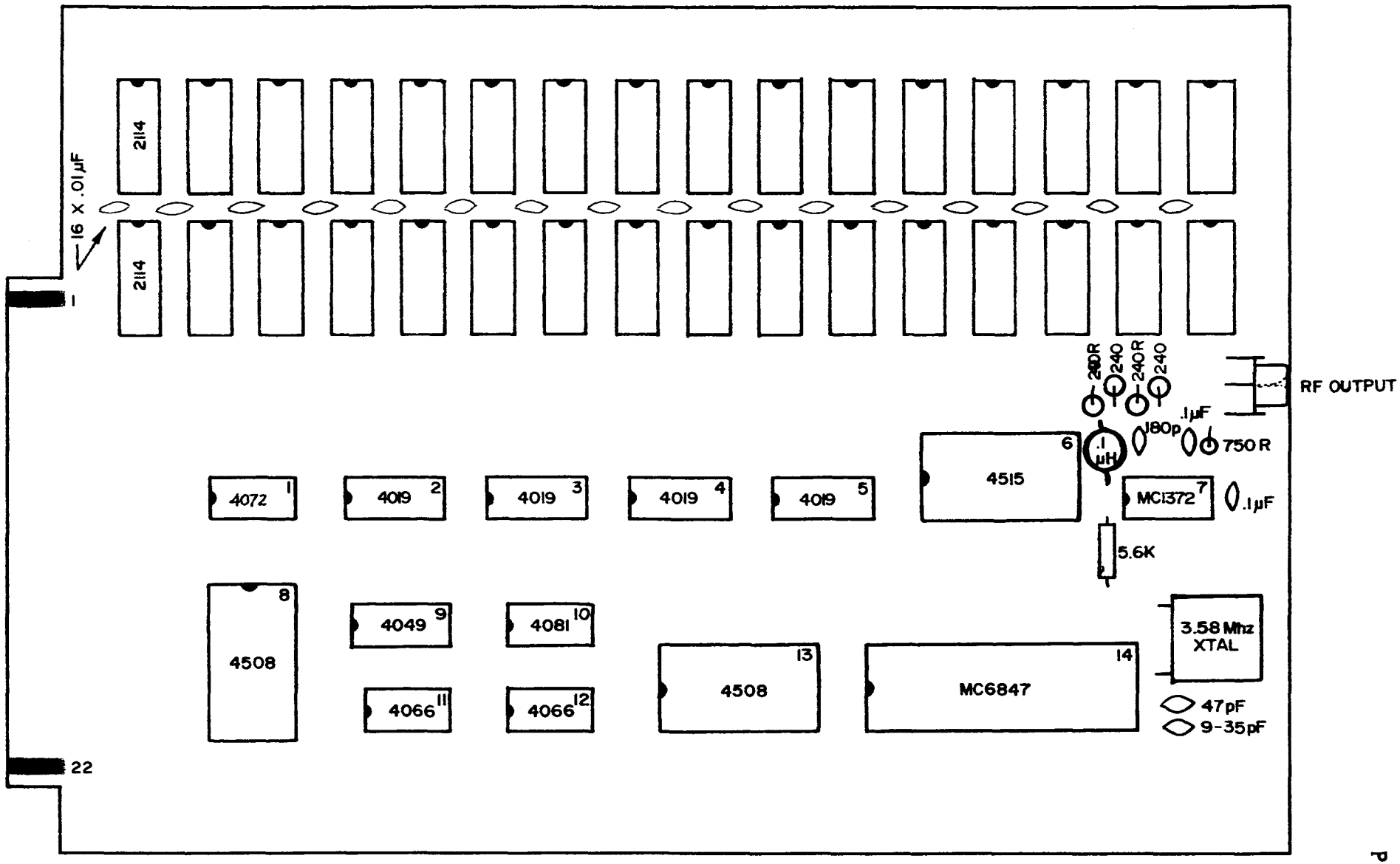
The colour video board which was described in issue 13 (as a wire-wrap project) is now available as a printed circuit board.

This circuit uses the Motorola MC6847 chip, and can be used as a high-density (256 x 192) graphics display, a 32 x 16 alphanumeric display with intermixed 8 colour semigraphic mode, or any other graphic mode in between. The board contains an on-board colour video RF modulator which outputs to any TV. There are 27 different modes of operation, and any one of them can be called under software control. This means that you can mix high density graphics with the alphanumeric mode to give you access to very sophisticated games. The mode control is memory-mapped at FF00. The video display starts at E000, which makes the display compatible with the Quest 12K BASIC. If you have a BASIC interpreter (12K or 6K version), then you can use all the readily available game programs which use memory mapped displays.

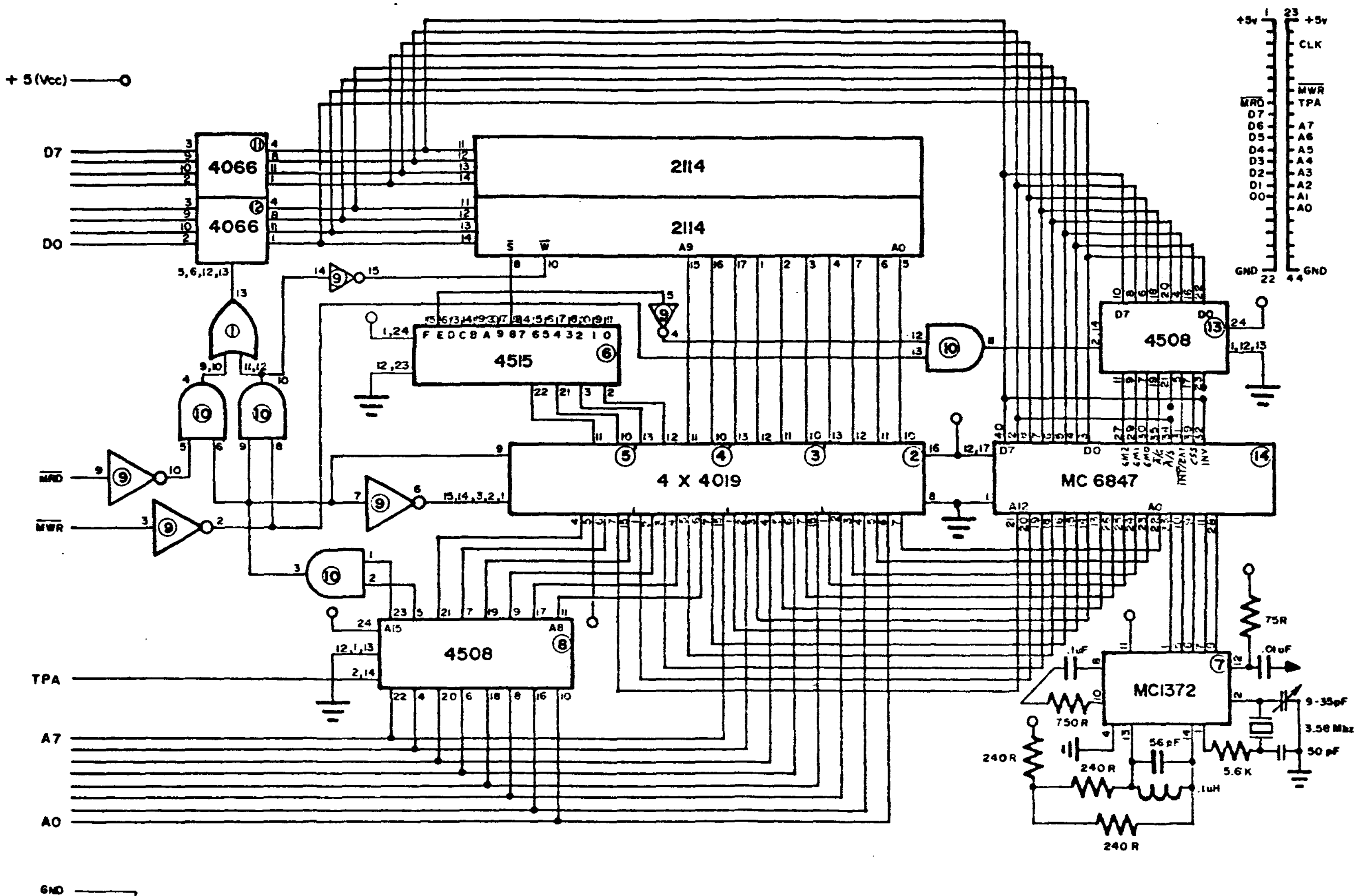
In addition to all the above-mentioned features, there is another. The board can hold a total of 16K bytes of static RAM. The RAM starts at C000 and ends at FFFF. The memory uses 2114 RAMs. The board is the same size as Tektron boards, and uses the Tektron bus (and also fits the new A.C.E. backplane).

The price is $35.00 Canadian. (U.S. members see note below). The parts can be bought for under $60, so for under $100, you can get a fully expandable video display board. At this time only bare boards are available. Full kits may be available to local members at a later date if there is enough demand.

cont. on pg. 37



A-C-E VDU 1802 VERSION 2


+ 5 (Vcc)
D7
D0
4066
4066
2114
2114
4515
4 X 4019
MC 6847
4508
4508
MC1372
MRD
MWR
TPA
A7
A0
GND
+5v
CLK
D6
D5
D4
D3
D2
D1
A6
A5
A4
A3
A2
A1
75R
.01uF
.1uF
750 R
240 R
56 pF
5.6 K
.1uH
9-35pF
3.58 Mhz
50 pF

A-C-E VDU-1802

ANALOG OUTPUT BOARD

Tom Crawford

General Description:

This board was designed to meet my requirement for analog outputs suitable for experimenting with graphics on an oscilloscope. The board provides 3 Digital to Analog (D-A) converters. Two of them are 8 bit converters, the third is 4 bits. The output voltage range can be setup as either unipolar or bipolar, up to about +/-10V, by adjusting gain and offset resistors. (The 4 bit D-A has no provision for offset resistors.) As shown in the schematic, all 3 D-A's provide 0 to +5V output.

The board also provids blanking logic, which can be used with the 4 bit D-A for Z-axis (intensity) blanking for the time between outputting to one of the 8 bit D-A's, and outputting the other.

As part of the same 24 line I/O port used to output to the D-A's, there are 4 extra lines, which can be programmed as either inputs or outputs. In the future I plan to use these lines to provide handshaking with some vector generation hardware. Currently, I use them to read a joystick interface.

Finally, this board provides an extra 24 line programmable parallel I/O port, which can be accessed via a 24 pin DIP header on the board.

Figure I provides a block diagram of the Analog board. The board can be mapped inbto any contiguous 8 bytes in the memory-mapped I/O Page, by appropriate settings of the 5-pole DIP switch.

Just about any oscilloscope can be used for display purposes. A bandwidth of about 100KHz is adequate, and the scope can be AC or DC coupled. Two points are woth noting though. First, be sure the scope common is grounded, to the same ground as your computer. Some older scopes (such as my DTI scope) have the chassis common capacitively coupled to BOTH lines of the incoming 110V AC power, and NOT grounded. This generates a substantial ground current when it is connected to the computer ground. The second point concerns AC coupling, especially the AC coupling used for Intensity or Z-axis inputs on most scopes. This is simplly a capacitor with one end sitting at several hundred volts, and the other end hanging around waiting for you to hook it up to your computer. When you do so, the surge current into the capacitor burns out the output op amp on the Analog Board. (It can also give you a healthy shock!) The solution is to hook a 100K resistor between the scope's Z-axis input and ground BEFORE hooking up to your computer. This same rule applies to the X and Y axis inputs of AC coupled scopes (like mine).

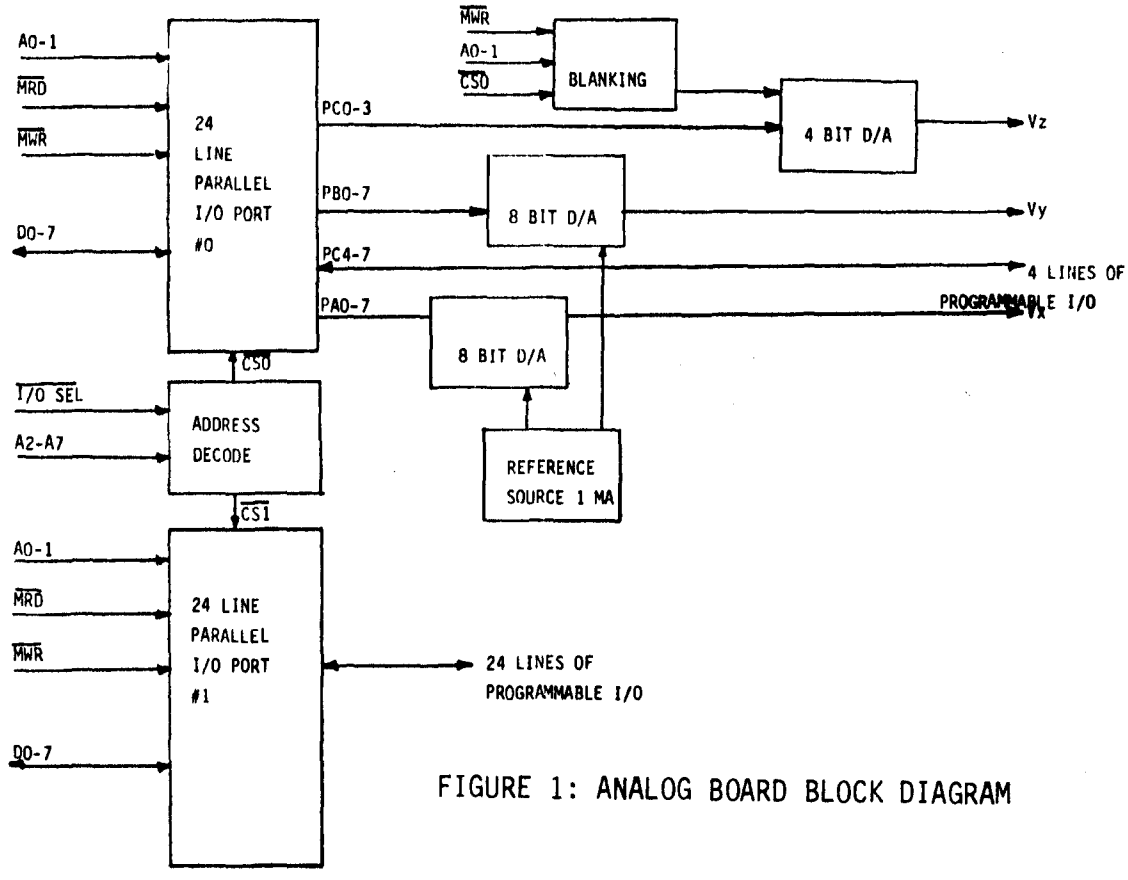
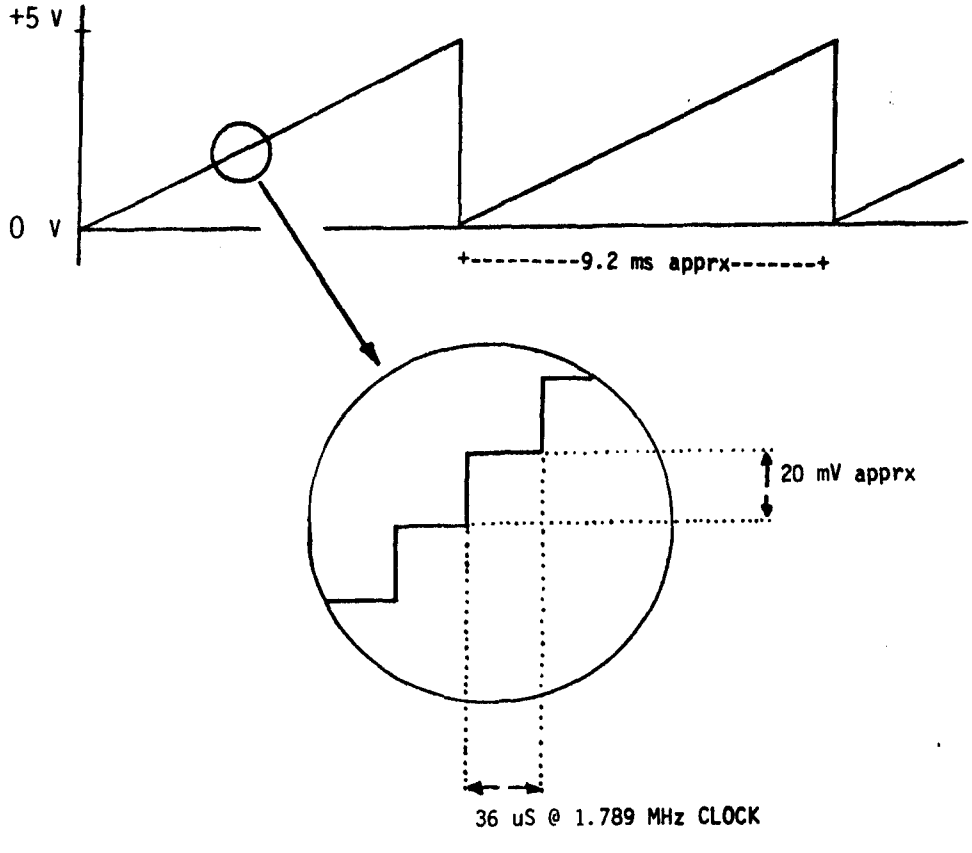
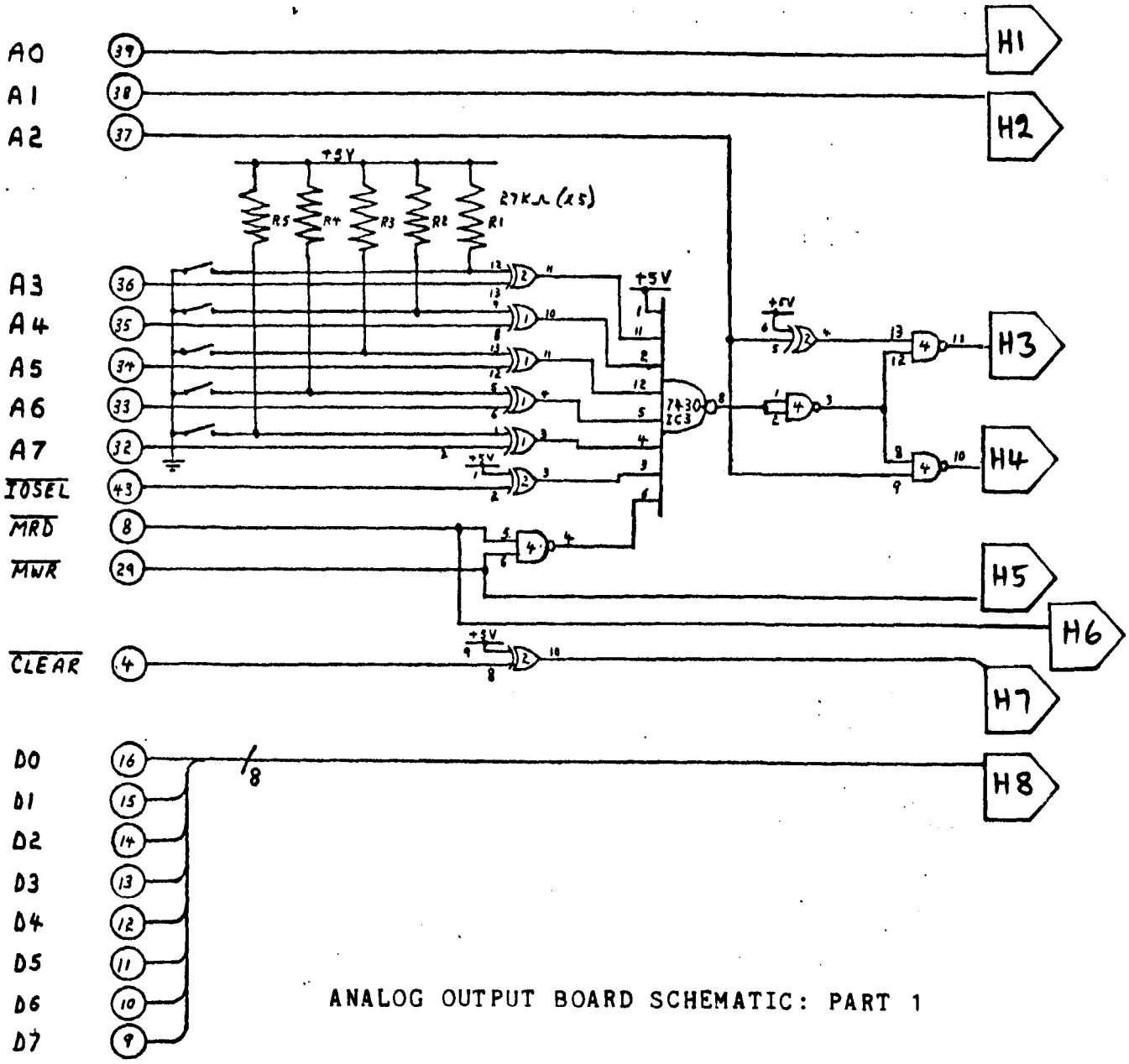


FIGURE 1: ANALOG BOARD BLOCK DIAGRAM

FIGURE 2: D/A CONVERTOR STAIRCASE WAVEFORM





ANALOG OUTPUT BOARD SCHEMATIC: PART 1

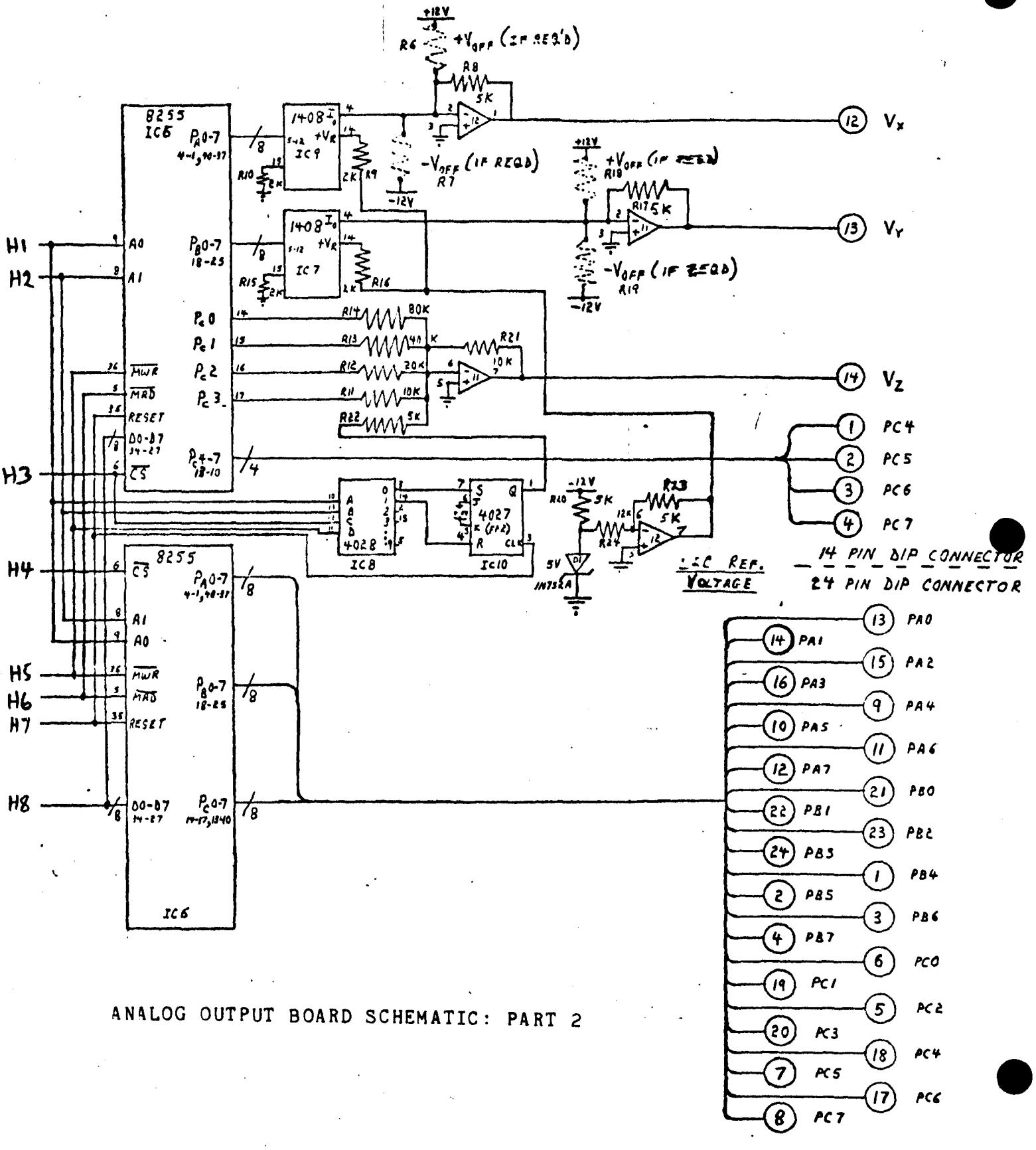
cont. from pg. 32

A full set of instructions will be included with each board. Delivery is four to six weeks. Send all orders to:

Bernie Murphy
102 McCraney St.
Oakville, Ont.
CANADA L6H 1H6

Note to U.S. members:

In the event that conversion of your money to Canadian currency is a problem (either in terms of convenience, or cost), please feel free to remit in the equivalent amount in U.S. funds. ($30.50 in the case of this board). This applies to all club offers, and to membership dues as well.



ANALOG OUTPUT BOARD SCHEMATIC: PART 2

Power Consumption

	+5V		+12V		-12V		
	TYP	MAX	TYP	MAX	TYP	MAX	(mA)
4030 (x2)	-	-	-	-	-	-	
4011	-	-	-	-	-	-	
7430	3	6	-	-	-	-	
8255	-	120	-	-	-	-	
1408L8 (x2)	27	44	-	-	15	26	
4028	-	-	-	-	-	-	
4027	-	-	-	-	-	-	
1458 (x2)	-	-	4.6	16	4.6	16	
TOTAL	-	150*	4.6	16	20	42	

* 270 mA with two 8255s installed

Address Assignments

ADDRESS	USE	
0	IC5, PORT A (Vx D-A)	
1	IC5, PORT B (Vy D-A)	
2	IC5, PORT C (C0-3 = Vz D-A)	
3	IC5 CONTROL PORT	
4	IC6, PORT A	Spare I/O Lines
5	IC6, PORT B	
6	IC6, PORT C	
7	IC6 CONTROL PORT	

Analog Output Board Parts List

IC1,2	4030	QUAD XOR GATES
IC3	7430	8-INPUT NAND GATE (TTL)
IC4	4011	QUAD 2-INPUT NAND GATES
IC5,6	8255	INTEL 24-LINE PROG. PARALLEL I/O PORT
IC7,9	1408L8	8-BIT DAC (CURRENT OUTPUT)
IC8	4028	BCD TO DECIMAL DECODER
IC10	4027	DUAL JK M/S FF WITH SET AND RESET
IC11,12	1458	DUAL 741-TYPE OP AMPS

D1	IN752A	5.6V ZENER DIODE, 400 mW
C1-9,12-16	0.01 ufd	CAPACITORS (POWER SUPPLY BYPASS)
C10,11	27 pfd	CAPACITORS
R1-5	47 K ohms	1/4 w RESISTORS
R6,7,18,19	OFFSET RESISTORS (OPTIONAL SEE TEXT)	
R8,17,20,22,23	5 K ohms	1/4 w RESISTORS
R9,10,15,16	2 K ohms	1/4 w RESISTORS
R11,21	10 K ohms	1/4 w RESISTORS
R12	20 K ohms	1/4 w RESISTORS
R13	43 K ohms	1/4 w RESISTORS
R14	82 K ohms	1/4 w RESISTORS
R24	12 K ohms	1/4 w RESISTORS
MISC:	24 PIN DIP CONNECTOR	
	14 PIN DIP CONNECTOR	
	5 POLE DIP SWITCH	
	PC BOARD	

External I/O Connector

i. 14 Pin DIP Connector

1	PC4, IC5 I/O LINE
2	PC5, IC5 I/O LINE
3	PC6, IC5 I/O LINE
4	PC7, IC5 I/O LINE
5	GROUND
6	+5V
7	N.C.
8	N.C.
9	N.C.
10	-12V
11	+12V
12	Vx (OUTPUT OF PORT A DAC)
13	Vy (OUTPUT OF PORT B DAC)
14	Vz (OUTPUT OF PORT C DAC)

ii. 24 Pin DIP Connector

1-4	PB4-7, IC6 I/O LINES
5	PC2, IC6
6	PC0, IC6
7	PC5, IC6
8	PC7, IC6
9-12	PA4-7, IC6
13-16	PA0-3, IC6
17	PC6, IC6
18	PC4, IC6
19	PC1, IC6
20	PC3, IC6
21-24	PB0-3, IC6

Software

General Description:

The 8255 programmable I/O port must be initialized to the desired configuration before it can be used. This is a very simple process, which involves writing a byte to the control port of the 8255 (A0=A1=1). The 8255 has numerous I/O options available; the reader is referred to the Intel Data Catalog (1978, pages 12-76 to 12-94) for further details.

For our purposes, it is necessary to setup the 8255 for Mode 0 I/O, with Port A,B and C (low) set for outputs. Port C (high) can be set for either outputs or inputs, since it is spare. Control byte #80 sets all the ports as outputs, while control byte #88 sets C (high) as inputs, and all the rest as outputs. Assuming the board is mapped into the low end of the I/O Page, the following code will perform the intitialization of the 8255 used for the D-As:

```
LDI  #FF
PHI  RE          ;POINT TO I/O PAGE
LDI  #03
PLO  RE          ;POINT TO 8255 CONTROL PORT
LDI  #88
STR  RE          ;INIT. THE 8255
```

Testing

Prior to testing the board, it is assumed that it has been assembled as follows:

- -all three-hole jumpers soldered in place and checked for contin and shorts, as applicable (this assumes the board is not plated through)
- -all resistors, capacitors and diode D1 are soldered into place (except offset resistors)
- -all IC sockets are soldered into place, if used
- -the 5-pole DIP switch is installed

A-Buss Interface Checks

The first stage in testing the completed Analog Output Board is to check out the address decoding and control buffering logic. This involves installiing only IC number 1,2,3 and 4 on the board, then plugging it into the system buss. Using variants of the program loop in Listing 1, exercise the board and test for proper board select (IC3, pin 8) and chip select signals (IC4, pins 10 and 11), using a logic probe or preferrable an oscilloscope. Try different settings of the address select switches on the board, in combination with

different low byte values in RE. Check that the control signals used on the board (MRD,MWR,A0,A1) all appear correctly. Since there are no data buss buffers on this board, no further checks of the buss interface are necessary.

B-I/O Port Checks

Install IC5 on the board. Using the program in Listing 2, exercise the I/O Port. Check that all 24 lines (mapped as outputs for this test) appear as both a 0 and a 1. If a problem appears, then modify the test routine to exercise only the bad lines and their adjacent pins and PCB runs in different ways. The problem is most likely an open or shorted connection on the PCB and can be quickly tracked down by observing the pattern of the errors produced. Be sure also the test the I/O lines right at the point where the IC pins go into the IC package, as well as on the associated PCB connections.

If IC6 is to be used, install it now and repeat the above test. Be sure to add 4 to the low byte of all I/O addresses used in Listing 2, so as to address the correct IC.

Listing 1- Address Decoding and Read Check:

```
          LDI    #FF      ;SETUP RE TO POINT TO MEMORY-MAPPED
          PHI    RE       ;  I/O PAGE,
          LDI    #00      ;            AND ANALOG OUTPUT BOARD
          PLO    RE
LOOP1:    LDN    RE       ;READ THE BOARD REPEATEDLY
          BR     LOOP1
```

Listing 2- 8255 I/O Port Output Test

```
          LDI    #FF
          PHI    RA       ;SETUP RA TO POINT TO MEMORY-MAPPED
          LDI    #03      ; I/O PAGE
          PLO    RA       ;          AND ANALOG OUTPUT BOARD,
          LDI    #80      ;          I/O PORT CONTROL BYTE
          STR    RA       ;INIT. I/O PORT AS ALL OUTPUTS
LOOP2     LDI    #02      ;POINT TO PORT C
          PLO    RA
          LDI    #FF      ;STORE FF INTO
          STR    RA       ;              PORT C,
          DEC    RA
          STR    RA       ;              PORT B,
          DEC    RA
          STR    RA       ;              AND PORT A.
          LDI    #02      ;POINT TO PORT C AGAIN
          PLO    RA
          LDI    #00      ;STORE 00 INTO
          STR    RA       ;              PORT C,
```

```
DEC     RA
STR     RA      ;               PORT B,
DEC     RA
STR     RA      ;               AND PORT A.
BR      LOOP2   ;GO DO IT AGAIN, SAM.
```

D/A Converter Checks

Install IC12. Using a DC voltmeter, check pin 7 of IC12. It should read approximately +2V. This is the reference voltage for the D/A Converters. Its absolute value isn't important; it must be stable, however. In case of problems, check D1, R20, R23 and R24. Also check that the same voltage appears at pin 14 of the sockets for IC9 and IC7.

Install IC9, the D/A converter chip connected to Port A of IC5. Do not install any offset resistors yet (R6,R7). Load the test program in Listing 3, and execute it. This will cause the D/A to generate a staircase waveform, at IC12 pin 1, covering its entire output range, and including every step. Check this wavefom carefully using a scope, to ensure it's correctness. It should look approximately as in Figure 2. If the staircase pattern is not as shown, look for a pattern which might signify which of the 8 input bits could be bad. Check the 8 input bits all look OK, then check the op-amp which is used to convert the output of IC9 from a current to a voltage. Remember that IC12 pin 2 is a virtual ground, and hence should be at ground potential!

Once the first D/A is working, install IC7 and IC11, and check out the Port B D/A. Remember to change the address from Port A in the test program in Listing 3.

Finally check out the 4 bit converter connected to Port C (low). Note that the staircase produced will have only 16 steps, instead of 256 and that the output ranges from 0V to about -10V.

Blanking Logic Check

If blanking of Vz is required between setting new values of Vx and Vy, then install IC8 and IC10. The blanking mechanism simply forces Vz into its negative voltage limit whenever IC10 pin 1 (Q output of FF2) is at +5V. This occurs between outputting to Port A, and outputting to Port B of IC5.

Listing 3--D/A Converter Staircase Test

```
LDI     #FF
PHI     RA      ;SETUP RA TO POINT TO MEMORY-MAPPED
LDI     #03     ;       I/O PAGE
```

```
        PLO     RA      ;       AND ANALOG OUTPUT BOARD
        LDI     #80     ;(I/O PORT CONTROL BYTES)
        STR     RA      ;INIT. I.O PORT AS ALL OUTPUTS
        LDI     #00     ;POINT TO PORT A
        PLO     RA
LOOP3   GLO     RB      ;GET REGISTER VALUE
        STR     RA      ;       OUTPUT IT
        INC     RB      ;INCREMENT THE REGISTER
        BR      LOOP3   ;       AND KEEP DOING IT
```

Simple Applications

A-Introduction

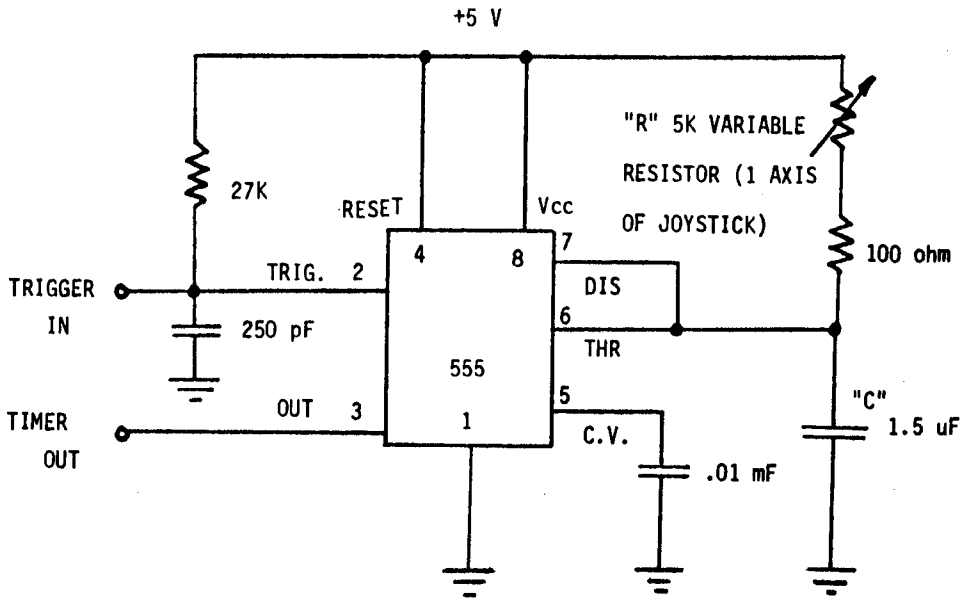
As a demonstration of the capabilities of the Analog Output board, I have put together a simple application. This consists of a small cross-hairs type of cursor, which can be moved around on an oscilloscope screen using a surplus 2-axis joystick. The oscilloscope's X and Y inputs are driven from the Vx and Vy outputs of the analog board. The joystick is read using 2 of the spare Port C bits. These are set up as inputs and connected to some simple interface circuitry. The software presented here is modular. It consists of a joystick position reading subroutine and a sub-picture display subroutine. Either of these can be used to interface the joystick or the scope display to other applications with virtually no changes.

B-Joystick Interface

The joystick interface is simply a pair of 555 timers wired up as one-shots. The variable resistance elements of the joystick are used as the resistances in the timer RC network. Therefore, when the timers are triggered, the durations of the resulting pulses are proportinal to the position of the joystick. The timers are calibrated so that the maximum pulse width corresponds to about 250 counts of a software counting loop. The timer circuitry I used is shown in Figure 3. A subroutine to read the position of a selected axis of the joystick is given in Listing 4. As noted in the Notes to Figure 3, I used the CS signal to IC6 to trigger the timers.

There are a few tricks involved in the coding of the joystick read subroutines. The first one concerns the fact that the subroutine only reads one axis at a time. It is therefore necessary to

FIGURE 3 -- JOYSTICK INTERFACE CIRCUIT



Notes: (1) 250 pf on Trigger line is to filter transients which were causing false triggering.
(2) 1.1 RC should be about 7.5 mS in order to provide a full-scale output of about 250 with the recommended read routine at Clock = 1.789 Mhz.
(3) Build 2 identical circuits, one for each joystick axi
(4) A suggested trigger for both circuits is CS for IC6 (pin 6) on the Analog Board.
(5) Connect the X-axis timer output to PC7 and the Y-axis timer output to PC6 for use with this demonstration, as written.

E R R A T A

In 'Building a Better BASIC', there were three segments of code which were incorrect.

Address 00E8 should be C1.
Address 0202 should be D4.
The seven bytes beginning at 14B4 should be
23 A0 0D 21 A0 EE C4.

>

call the subroutine twice in succession in order to get the X and Y positions of the joystick handle. Since all the timers are triggered every time any one of them is to be read, then it is possible that, say, the Y-axis timer is still timing after the X-axis timer has finished. The next call to the joystick read routine, to read the Y-axis timer, would return an incorrect value, unless the routine checks to ensure that the Y-axis timer is off before it is triggered. This is done in the read routine in Listing 4.

It is also necessary to ensure that the timer has been off a minimum amount of time (about 300 micro-seconds in my case), to allow the timing capacitor to be completely discharged by the timer's Reset line. Otherwise an incorrect value will again result. This is also done in the read routine in Listing 4.

LISTING 4 -- JOYSTICK READ ROUTINE

1. Pass mask in D, with set bit indicating which timer to read.
2. Result (0-255) passed back in D.
3. Register RA.1, RB.1 must point to I/O Page. Registers RA.0 and RB.0 are destroyed.
4. Register RC.0 is destroyed (used as timer counter).
5. IC6 CS is assumed to be the timer's trigger.
6. SCRT mechanism, modified to save D-register , is used for CALL an RETURN.

```
READR:  STR     R2              ;SAVE MASK
        LDI     #02             ;
        PLO     RA              ;INITIALIZE RA (INPUT)
        LDI     #04             ;
        PLO     RB              ;AND RB (TIMER TRIGGER)
READ5:  LDN     RA              ;
        AND                     ;ENSURE TIMER IS OFF,
        BNZ     READ5           ;
        LDI     #0E             ;AND FULLY RESET
READ10: SMI     #01             ;
        BNZ     READ10          ;
        PLO     RC              ;INITIALIZE COUNTER TO ZERO,
        LDN     RB              ;AND START TIMER
READ15: LDN     RA              ;
        AND                     ;TIME IT.
        BZ      READ20          ;DONE YET? YES.
        INC     RC              ;NO.
        BR      READ15          ;KEEP GOING
READ20: GLO     RC              ;PUT RESULTS IN D,
        RETURN                  ;AND RETURN
```

C-Sub-Picture Display Subroutine

The Analog Output Board, when used as an X-Y graphics display on a scope, provides the capability of displaying more than 65,000 individual points. To provide storage for the ON/OFF state of each of these points would equire 8KB of memory, plus software to do the mapping between bits in memory and points on the screen. This would also require a storage scope, since it would take many seconds to output the contents of memory to the display just once.

For many graphics applications, most of the bits in memory are OFF, since most of the display screen points are not lit. This would result in inefficient use of memory, and of the time required to output all those OFF bits. An alternative technique is to store only the X and Y values of the points which are to be ON. Although it now takes 16 bits of memory to describe each point in the list, substantial memory savings occur for displays with only a few percent, or less, of the display points turned ON. There is also an execution time saving since no mapping of stored information to display points needs to be done. The data bytes in memory are output directly to the Analog Output board, as X and Y data values. It is also possible to use and ordinary scope as a display device, since outputting the list can be done repetitively, fast enough to provide a visually continuous display.

When considering the movement of images in the display, however, another problem becomes apparent. Whenever an image is to be moved, it is necessary to stop the display, calculate the new values of all the X and Y points in the display list, then resume displaying, with the image now in the new position. This is a cumbersome, time-consuming process.

A simpler way to provide moveable images involves the concept of a "local origin". This is not the same as the display origin at (X,Y)=(0,0). In fact, it is not directly related at all to the analog output display. Instead, it is related to the image, or "sub-picture" which it provides an origin for. Figures 4 and 5 illustrate the difference between the display origin and a local origin.

Several ways have been developed to describe a sub-picture in the form of a display list which is relative to a local origin. After due consideration of the impact of these methods on the software required for such future enhancements as sub-picture rotationa and zoom, I elected to use X-Y pairs of 2's complement 8 bit integers. This is illustrated in Figure 5, where the end-points of the cross hairs are labelled, relative to the local origin. Table 1 provides the display list for the cross hair sub-picture to be used in this demonstration. The display list is simply a list of X-Y pairs of 8-bit integers, which the co-ordinates of the points to be displayed, relative to the local origin. The list is terminated by 80 hex (-127 base 10) in the byte normally occupied by the next X co-ordinate. Therefore every display list should contain an odd number of bytes.

The sub-picture display list is related to the display origin at the time it is output to the display. This provides for simple, fast movements of sub-pictures around the display -- an important requirement for dynamic displays. The subroutine to output the sub-picture at a selected point on the display is given in Listing 5. A feature of this routine is that it will always move the beam to the required display point before outputting the sub-picture and will move the beam back to the display origin (0,0) afterwards. This is required for AC coupled scopes (like mine).

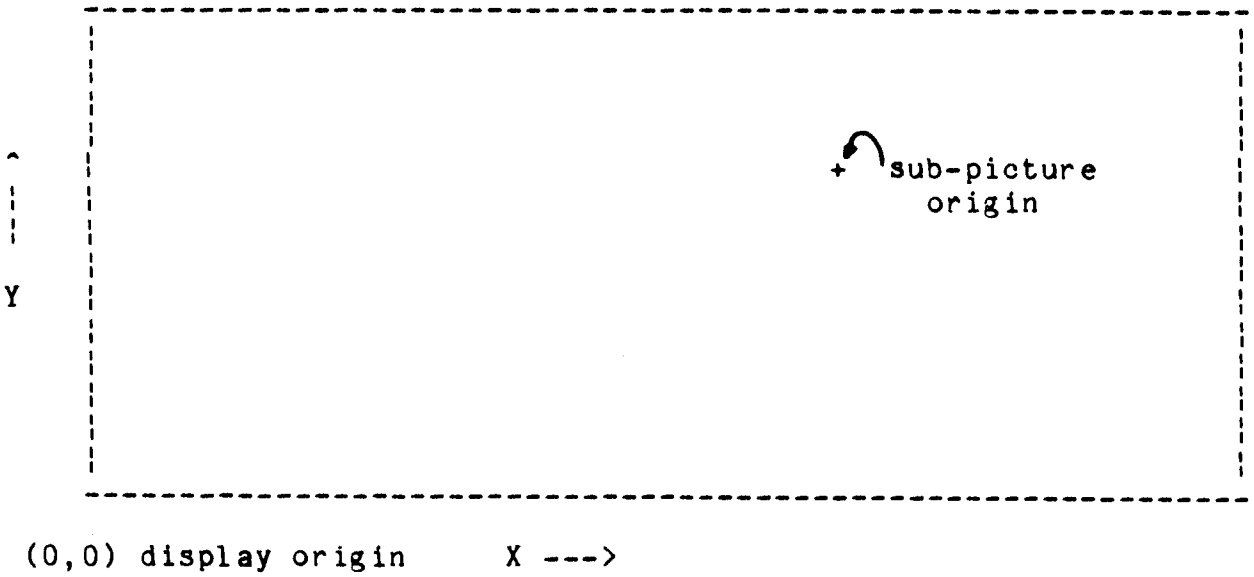


FIGURE 4: Sub-Picture Relative to Display Origin

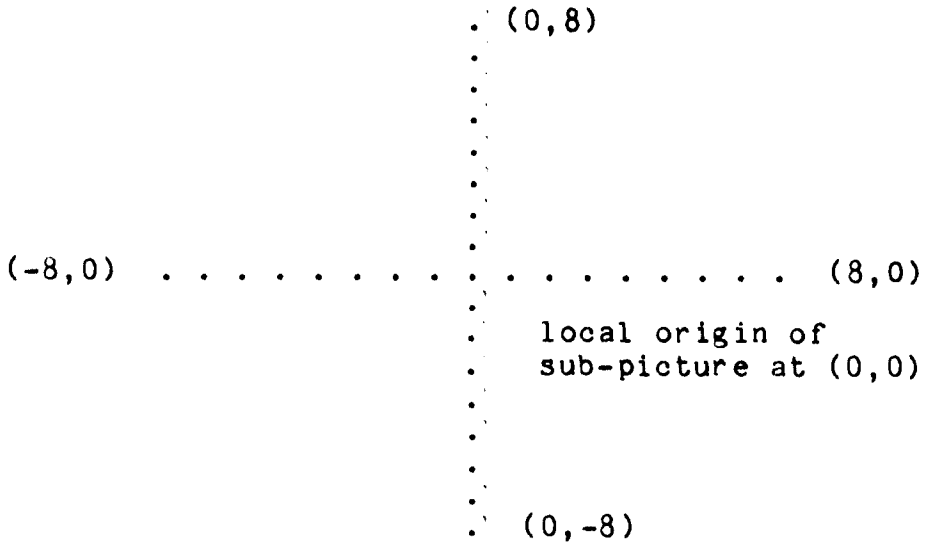


FIGURE 5: Sub-Picture Relative to Local Origin

TABLE 1: Cross-hair Sub-Picture Display List

```
CRSHRS:   .BYTE   #F8,#00,#FA,#00                 ;LEFT ARM
          .BYTE   #FC,#00,#FE,#00,#00,#00
          .BYTE   #00,#F8,#00,#FA                 ;LOWER ARM
          .BYTE   #00,#FC,#00,#FE,#00,#00
          .BYTE   #08,#00,#06,#00                 ;RIGHT ARM
          .BYTE   #04,#00,#02,#00,#00,#00
          .BYTE   #00,#08,#00,#06                 ;UPPER ARM
          .BYTE   #00,#04,#00,#02,#00,#00
          .BYTE   #80                             ;LIST TERMINATION
```

NOTE: Every second display dot used.

DISPLY--SUB-PICTURE DISPLAY ROUTINE

1. RA.1 and RB.1 must point to I/O page. RA.0 and RB.0 are destroyed.
2. RC must point to start of display list. List terminator is X=80 H.
3. RD must point to the display origin of the sub-picture in 8-bit absolute values.

```
DISPLY:   LDI     #00             ;INITIALIZE
          PLO     RA              ;RA TO POINT TO X OUTPUT
          LDI     #01             ;
          PLO     RB              ;AND RB TO Y OUTPUT
          SEX     RD              ;SET X TO RD
          LDXA                    ;
          STR     RA              ;MOVE CRT BEAM TO LOCAL ORIGIN
          LDX                     ;
          STR     RB              ;
          DEC     RD              ;RESTORE RD, AND START DISPLAYING
DISP5:    LDN     RC              ;THE SUBPICTURE
          XRI     #80             ;
          BZ      DISP10          ;DONE YET? (X=80) YES.
          LDA     RC              ;.NO. ADD X VALUE TO ORIGIN,
          ADD                     ;
          IRX                     ;
          STR     RA              ;AND OUTPUT IT
          LDA     RC              ;ADD Y VALUE TO ORIGIN,
          ADD                     ;
          STR     RB              ;AND OUTPUT IT.
          DEC     RD              ;
          BR      DISP5           ;GO BACK FOR NEXT POINT
DISP10:   LDI     #00             ;MOVE CRT BEAM BACK TO (0,0)
          STR     RA              ;  (FOR AC-COUPLED SCOPES),
          STR     RB              ;
          SEX     R2              ;RESTORE RX,
          RETURN                  ; AND RETURN
```

LISTING 5:SUB-PICTURE DISPLAY ROUTINE

MOVEABLE CURSOR ROUTINE

Listing 6 combines the joystick reading subroutine and the sub-picture display routine withe some initialization logic, to actually produce the moveable cross hairs display on an oscilloscope. Some possible improvements to this routine might include:

- -entering it via a 30 Hz or 60 Hz interrupt, to provide a constant refresh rate, and hence a constant intensity display
- -addition of "turtle" logic to allow the cursor to leave a trail behind it as it is moved around the screen
- -addition of a straight-line calculator routine to allow the generation and display of a straight line between 2 points selected using the cursor and an additional pushbutton wire into PC5.

CURSOR- Cursor Display Centred at Joystick Position

1. This routine reads the joystick, displays the cursor and repeats.
2. Cursor location (from joystick) is stored at 01FE (X) and 01FF (Y), pointed to by RD.
3. Cursor pattern (sub-picture) is stored at 0200. RC points to it.
4. RA and RB are used as I/O Page pointers.

```
          ORG    #0100            ;ASSEMBLE AT 0100
CURSOR:   LDI    #FF              ;INITIALIZE RA.1 AND RB.1 TO POINT
          PHI    RA               ;TO I/O PAGE
          PHI    RB               ;
          LDI    #03              ;POINT TO I/O PORT CONTROL BYTE,
          PLO    RA               ;
          LDI    #88              ;AND INITIALIZE IT
          STR    RA               ;
CURS5:    LDI    CRSTOR.1         ;
          PHI    RD               ;POINT RD AT CURSOR STORAGE LOCATIONS
          LDI    CRSTOR.0         ;
          PLO    RD               ;
          LDI    #80              ;GET X CO-ORDINATE,
          CALL   READR            ;
          STR    RD               ;AND SAVE IT.
          INC    RD               ;
          LDI    #40              ;GET Y CO-ORDINATE,
          CALL   READR            ;
          STR    RD               ;AND SAVE IT.
          DEC    RD               ;RESTORE D.
          LDI    CRSHRS.1         ;
          PHI    RC               ;POINT RC TO DISPLAY IT,
          LDI    CRSHRS.0         ;
          PLO    RC               ;
          CALL DISPLY             ;AND GO SHOW IT AT CURSOR POSITION
          BR     CURS5            ;GO DO IT ALL AGAIN.
CRSHRS    .EQL   #0200            ;LOCATION OF CROSSHAIRS DISPLAY LIST
CRSTOR    .EQL   #01FE            ;LOCATION OF CURSOR STORAGE (X,THEN Y)
```

LISTING 6:Moveable Cursor Routine

>